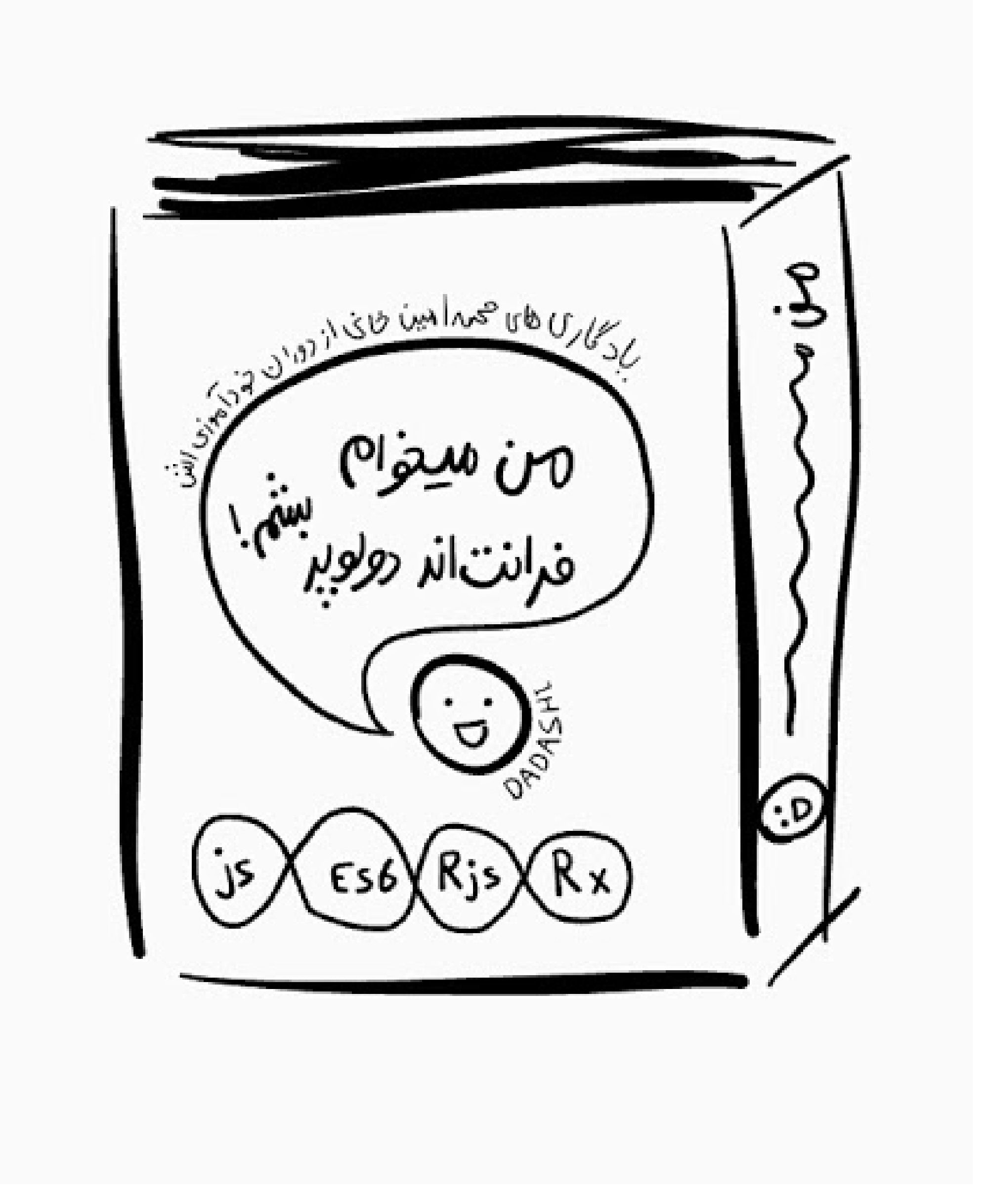
یادگاری Reactjs

داداشی | khanim۹۷@gmail.com

اسفند ۹۷ تا مهر ۹۸



فهرست مطالب

فصل ۱

گیت Git

پیشگفتار داداشی سلام داداشی! الان که این کتابو مینویسم برای بازنویسی همان پروژه های react، redux با git هست. جزو چیزایی هست که هر برنامه نویس چه بک چه فرانت باید بلد باشه! جزو تعلیمات پیش فرض است. این لینک دانلود این دوره اموزشی:
http://bitdownload.ir/lynda-up-running-git-github/

۱.۱ کنترل ورژن با git

نصب
https://git-scm.com

۱.۱.۱ ایجاد repo و فایل

گیت یک سیستم کنترل ورژن است. وقتی نصب شد در bash مینویسیم نه ویندوز پاورشل. بیایید یک فایل (repository) ایجاد کنیم. در ان دو فایل هم ایجاد کرده و اماده تغییر نوشته های داخلش شویم.

```
$ cd Desktop
$ git init dadashi
$ cd dadashi
$ touch index.html
$ touch about.html
clear
$ git add .
$ git status
```

در دوخط اخر همه فایل ها را سلکت کرده و به خودمان نشان دادیم. اگر میخواستیم فقط index.html را سلکت کنیم:

```
$ git add index.html
$ git status
```

وارد کردن اطلاعات

```
$ git config --global user.name "dadashi"
$ git config --global user.email "khanim97@gmail.com"
```

بریم اطلاعات خودمو ببینم و اخرین تغییرات:

```
$ git log
```

برای بالا اوردن bash میتوان کلیک راست کرد روی پوشه یا در ترمینال از cd استفاده کرد.

۲.۱.۱ بازگشت تغییرات با checkout

اگر در برگه html تغییر بوجود بیاریم (با ویرایشگر کد مثل اتم) بعد از checkout استفاده کنیم تغییرات ما undo میشه!

```
$ git checkout index.html
```

۳.۱.۱ ارور و راه حلش

Another git process seems to be running in this repository

```
rm -f .git/index.lock
```

۴.۱.۱ کد status و add

اگه تغییراتی بوجود بیارم در برگه انگاه بزنم status قرمز میاره همون برگه رو حالا add میزنم و دوباره status میبینم سبز نشون میده!

```
$ git status
$ git add .
$ git status
```

۵.۱.۱ حذف فایل

دو راه برای حذف کردن یک فایل
راه یک:

```
$ git rm index.html
```

حذف دستی و سپس وارد کردن کد:

```
$ git add index.html
```

۶.۱.۱ بازگشت حذف

```
$ git reset HEAD index.html
$ git checkout index.html
```

۷.۱.۱ شاخه branch

یادته لیندا فایل های تمرینشو؟ هرکدومشون یه شاخه هستن! به همین سادگی!
کدهای زیر به ترتیب برای ایجاد شاخه، دیدن شاخه ها، حذف شاخه، تغییر نام شاخه، همانند کردن شاخه کنونی با شاخه دیگر و اخرین کد برای تغییر مسیر به شاخه ای دیگر است.

```
$ git branch app01
$ git branch
$ git branch -D app01
$git branch -m app01 app1
$ git merge app01
$ git checkout app01
```

۲.۱ شبکه اجتماعی GitHub

به به! دنبال همین بودم! میخواستم برم یه پروژه اماده از گیتهاب بیارم! شبکه اجتماعی برنامه نویساس! این کتابچه هم راهنمای کار با پیچیدگی های ارتباط برنامه نویسا با هم دیگس

```
$ cd Desktop
$ git clone https://github.com/...
// follow the instructions from author
```

۱.۲.۱ فقط یک شاخه گل رز لطفا :)

الکی مثلا من توی گلفروشیم دارم گل میخرم برم خواستگاری! از هرچه بگذریم سخن دوست خوش تر است.

اگر بخوام یه branch از یک repository از گیتهاب بیارم توی دسکتاپم چیکارم کنم؟ یا master میارم بعد یه شاخشو فراخوانی میکنم یا از همون اول فقط همون شاخه رو میارم. اگرم تمام شاخه نبات هارو بخوام میرم از سومین قطعه کد استفاده میکنم.

```
git branch -a
// see all branches
git checkout -b dadashiBranchName origin/02_01
// remotes/origin/02_01
```

```
cd Desktop
git clone -b 00_start https://github.com/...
// 00_start from github
```

```
mkdir dadashiDirection
cd dadashiDirection
git clone --mirror https://github.com/.....git .git
ls
git config --bool core.bare false
git reset --hard
```

۲.۲.۱ پروژه خودم شود!

به قطعه دوم کد بالا نگاه کن داداشی! برای چی اوردیمش؟ که یه شاخه خاص بیاریم! اونم شاخه شروع start ! درست میگم داداشی؟ خب حالا میخوایم اون فایلی که git. داره داخلش و اگه ما بریم بزنیم git-log مال اون نویسنده قبلی میاد ما نویسندش بشیم و لاگ ما بیاد! (من نوشتم لاگ تو که نخوندی لاک؟ ها؟ من پسرما دختر نیستم داداشی!!) حالا به ادامه قطعه کدهای شماره دو اینو اضاف کن:

```
cd ...
ls
rm -dfr .git
git log
// khli shod ;)
git init
```

۳.۲.۱ در github تمام فایل های بارگذاری نمیشوند!

منظور اخرین خط همون تصاویره! میخواد بگه تصاویر به گیتهاب نمیان! شایدم گیتهاب برای کنترل حجم هاست سایتش یا برای تمرکز ما بر خود پروژه این اجازه رو نمیده! گفته بیخیال بابا! بیخیال عکسا!! عکس میخوای چیکار؟ ما خودمون یه پا شکل و شمایلیم! ماشالا ماشالا!!

- .DS_Store
- node_modules
- .tmp
- builds/development/images/**

فصل ۲

به اسکریپت خوش آمدید!

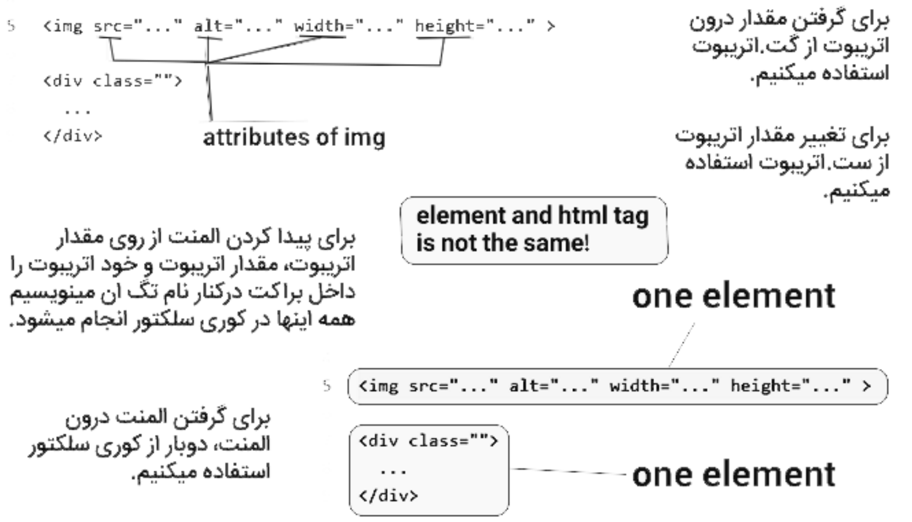
۱.۲ پیشگفتار

درود! من محمدامین هستم. در این کتاب بصورت شلخته اموزی رفتم و از ویدیوهای لیندا ان قسمتی که دوست داشتم یا نیاز داشتم را برداشتم اوردم. امیدوارم از خواندن این متن لذت ببرید! :-)

لازم نیست همه را حفظ کنید! از این بیشتر هم بود اما نمیخواداد! لازم نداریم! هنگام اجرا همین ها زیادی هم هستند! :)

برای راحتی کباب ساختم. یعنی ۸۰ ویدیو لیندا اسکریپت را به ۴ قسمت ۲۰ تایی تقسیم کردم. ۲۰ ویدیو اول را با نام ربع اول در یک پوشه قرار دادم. خیلی سریع تمام شد! به این ترتیب یادگیری خودم را اندازه گرفتم و شاد شدم! اگر ویدیو ها را همینطوری نگاه میکردم جشن ربع اول را نمیگرفتیم و انرژی برای باقی راه نداشتیم! مگه نه؟ پس بزن بریم! نمیخواهم فقط درس بدهم! باید احساسات همسو شود. من موفق میشوم انرژی به شما برسانم و شما را تا پابان با کوله باری از دانش کاربردی برنامه نویسی به زبان اسکریپت همراه خود بکنم! ها ها ...! دستتو بده من! حالا بپر توی دنیای اسکریپت! جاوا اسکریپت!

منابع خوب برای جاوا اسکریپت خام را یاد گرفتن سایت javascript.info هست که TOC داره و میشه گزارش ویدیویی با اسکرین رکوردر گرفت و بعدا دید که تشویق شد برای دوباره سراغش رفتن. این اطلاعات، پیش نیاز است.



۲.۲ پیشنیاز

برگه index.html را میسازیم بعد برگه script.js را در آن فراخوانی میکنیم. چنانچه در بالای صفحه فراخوانی کنیم مثل این:

```
<script type="text/javascript" src="js.js" ></script>
</head>
```

مشکلی که پیش میاد اینه که المنت ها در بادی تعریف میشن یعنی بعد از این خط کد! و دیگه دستش به المنت ها نمیرسه! انگار المنت ها تعریف نشدن! بنابراین بهش defer اضافه میکنیم...

```
<script type="text/javascript" src="js.js" defer></script>
</head>
```

حالا قضیه حله :)

۱.۲.۲ راه دوم: فراخوانی در کف بادی

اینطوری هم میشد فراخوانی کرد اگه نمیخواید از defer یا async استفاده کنید! خودمم راه دومو انتخاب میکنم!

```
<script type="text/javascript" src="js.js"></script>
</body>
```

۲.۲.۲ یک متغیر تعریف کنید!

```
var x = 2;
```

۳.۲.۲ نامگذاری با آندرلاین

همونطور که میبینید نامگذاری با آندرلاین درسته و خط کد دو اشتباه! :)

```
1 fn_pxl()   //true
2 fn-pxl()
```

۳.۲ آرایه

```
var x = new Array(3, 33);
x = [3, 33]
```

نکته ای که وجود دارد این است که برای تعریف ارایه همیشه لازم نیست از var استفاده کنیم. بله! براکت کار خودشو کرد!

۱.۳.۲ مهمانی خانم ارایه برای همه انواع متغیر ها

```
x = ['gold', "1", 1, null, true, false]
```

۲.۳.۲ بازسازی ارایه

به راحتی ارایه x را نادیده گرفته و مقدار جدید به ان میدهد.

```
x = new Array("purple");
```

۳.۳.۲ اندازه ارایه و یک عضو از یک ارایه

```
x.length   // 2
x[0]       // 3
```

۴.۳.۲ مقایسه کنید!

در هر دو خط کد زیر، x1 یک عدد است. عددی که اندازه ارایه را نشان میدهد.

```
x1 = x.push('gold');
x1 = x.length;
```

۵.۳.۲ حذف و اضافه کردن اعضای یک ارایه

متد push به اخر ارایه عضو اضافه میکنه! متد unshift هم از اول اینکارو میکنه! متد pop از اخر یکی کم میکنه و متد shift اولیو حذف میکنه! به همین سادگی! :)

```
x.push('gold', 2)
x.unshift('go', 2)
x.pop()
x.shift()
```

۶.۳.۲ متد splice چیکار میکنه؟

```
var months = ['Jan', 'March', 'April', 'June'];

months.splice(1, 0, 'Feb');

// output: Array ['Jan', 'Feb', 'March', 'April', 'June']

months.splice(4, 1, 'May');

// output: Array ['Jan', 'Feb', 'March', 'April', 'May']
```

more info: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

۷.۳.۲ از اخر به اول با reverse متد

```
x.reverse();
```

۸.۳.۲ متد join میگه بین اعضا چی باشه!

```
var x1 = x.join(" | ")  // x1 has joiner but x dosn't change at
    all.
```

۹.۳.۲ نمیدونم ایندکس اف چیکار میکنه!

indexOf

```
pens=["wonder", 234, null, true]
var result= pens.indexOf(null,2)
document.body.innerHTML="<h1>"+pens[result]+"</h1>"
//browser displays "null", true
```

۴.۲ خرده پروژه: یک ارایه جدید با یک عضو بیشتر بسازید!

```
pens = ["go",2]
var pens1 = pens.slice()
pens1.push("g")  //  ["go", 2 , "g"]
```

۵.۲ تفاوت را احساس کنید!

```
1 var z = document.querySelector("img").getAttribute("src");
2 var x = z.slice(4);
3 var y = z.slice(0,4);
```

خط دوم: از پنجمی تا اخری! اگر مقدار دوم ندیم تا اخرین کاراکتر (اخرین عضو ارایه) کپی میکنه!

خط سوم: از کاراکتر شماره صفر تا شماره ۳ (چهار کاراکتر اول) را کپی میکند.

۶.۲ نمایش در بروزر

```
  var date=new Date();
document.body.innerHTML="<h1>Today is:"+ date +"</h1>"
```

```
var a=10; var b=a+2;
if (a>b) {document.body.innerHTML="<h1>a is bigger! a:"+a+"<h1>"}
else {document.body.innerHTML="<h1>b is bigger! b:"+b+"<h1>"}
```

```
function fn1(a,b){document.body.innerHTML= a+b};
fn1(3,5)
```

```
function fn2(a,b){
// statement? true : false
  a > b ?  document.body.innerHTML = "<h1>TRUE</h1>" : document.body.innerHTML = "<h1> FALSE </h1>"
//note: it doesnt need 'return ...' at the end of difinition of fn}
//
fn2(3,4)
```

Here is function about boolean.
this fn is different from ”if” statement.

```
function fn3(a,b){
  var result;
  a>b? result=["First Fraction",a]:result=["Second Fraction",b];
  var d=document.body.innerHTML="<h1>the "+result[0]+" value is "+result[1]+" and it's the bigger one!</h1>"
  return d }
fn3(3,4)
```

2.7 Global & Local-variations

global variables are defined anywhere but local variables are defined just in an area like function!

let,var,const

```
const x0 = 99
 //this is global but dont change!
var x1 = 100
 //global variation.defined in all fn and out of functions area.
let x2 = 101 //not defined in function area
function fn1(){
  var x3 = 102 //not defined out of fn1, defined for fn2
  let x4 = 103 //not defined in fn2
  function fn2(){
    let x4 = 102
    return x3/x4  } // x3=x4 --> x3/x4=1
```

```
    return x4  }  //x4 = 103 ,  x4 != 102
//dsply
document.body.innerHTML= fn1()
```

x0 its Global.
x1 its Global.
x2 its Local. It means that defined for the area out of function
x3 its Global
first x4 is Local.
second x4 is Local, too.

فصل ۳

ربع دوم: شی، تابع و دام

۱.۳ یک object تعریف کنید!

```
var a1 = new Object();
a1 = {
  title: "lyna-js",
  author: "Mohammad Amin",
  age: 24,
  date_of_publishing: "98.05.21"
}
```

توجه اخرش ، داره بجای ؛
برای تعریف شی این راه هم وجود دارد:

```
var a1 = new Object();
a1.title = "lynda-js";
a1.author = "mhmd amin";
a1.age = 23;
a1.date = "98.05.21";
```

میخواید یکی از چیزاشو در بادی نمایش بدید؟ از این قطعه کد بهره ببرید!

```
document.querySelector("#id2 p").append(a1.date);
```

تعریف ابجکت و ارایه خیلی سریع: آرایه بدون نوشتن new-Array() هم قابل تعریف است با کروشه! همچنین ابجکت هم بدون new-Object() هم تعریف میشود با پرانتز!

۲.۳ یک ابجکت مشتی :)

این ابجکت تمام انواع متغیر را داراست!

```
var course = new Object();
course={
  title: "gold lesson",
  author: "capalood",
  published: true,
  date_published: "97.12.12",
  free: true,
  price: null,
  views: 0,
  updateView: function(){return course.views++}
  }
```

به کنسول لوگ بروید و بزنید course.views ببینید چه عددی نشون میده! سپس این دو خط کد را به برگه اسکریپت اضافه کنید.

```
course.updateView();
course.updateView();
```

دوباره به کنسول لوگ بروید و همین عبارت را وارد کنید تا ببینید همان عدد دوتا بیشتر شده!

۳.۳ به property های یک object چگونه دسترسی پیدا کنیم؟

در اینجا به title میگن پروپرتی!

```
course.title
course["title"]
course[ " title: 'gold lesson' " ]
```

بنظرتون فرق این دو تا چیه؟ دومین خط به یک متغیر اشاره میکنه! مثلا به اولین پروژه فصل ۴ بروید و ببینید چگونه از یک متغیر بعنوان پروپرتی یک ابجکت استفاده شده.

```
course["title"];
course[title];
```

باز بینی: اینجا اگر بخوایم یک متغیر را بعنوان پروپرتی قرار بدیم گیج میشیم کدومو قرار بدیم! همیشه اولی را به خاطر بسپاریم بهتره! نقطه دارش را بیخیال شویم.

```
course[title];
course.title ;
```

۴.۳ خرده پروژه: تابعی که ابجکت تولید میکرد object-constructor

```
function course (title, author, level, date, published, views) {
this.title= title;
//attention: use "=" not ":"
this.author= author;
//attention: use ";" not ","
this.level= level;
this.date = date;
this.published = published;
this.views = views;
this.updateView= function ( ) {return ++this.views}
 }
```

۱.۴.۳ ادامه: یک بچه ابجکت چطور به دنیا میاد؟

```
var course02 = new course( "book1", "shervin", "3", "97.12.12", true, 10 );
```

حالا اگه میخوای یکی به عدد ده اضافه بشه این دو خط را به برگه اسکریپت خودت اضافه کن! بعد به کنسول لوگ برو و تماشا کن پسر!

```
course02.updateView();
console.log(course02.views);
```

اینا چقدر شبیه همن!! مگه نه؟! :)
دوستای قدیمی هر چند سال یکبار دور هم جمع میشن و یه عکس یادگاری میگیرن! نگاشون کن! چه قشنگ شدن!

```
var x = new course();
var x = new Date();
var x = new Array();
var x = new Object();
```

۲.۴.۳ کارخانه تولید ابجکت که میگن اینه! بقیه اداشو درمیارن!

```
var courses=[
  new course("go0","sh0",true,10),
  new course("go1","sh1",true,11),
  new course("go2","sh2",true,12),
  new course("go3","sh3",true,13),
  new course("go4","sh4",true,14),
  new course("go5","sh5",true,15)   ]
```

تکون بده! حالا یه تکونی بخوایم به بچه دومیه بدیم چیکار کنیم؟ کافیه توی کروشه عدد ابجکت مورد نظرمونو وارد کنیم.

```
courses[2].updateView();
courses[2].updateView();
courses[2].updateView();
console.log(courses[2].views)
// it returns `15`
```

۵.۳ تابع عجیب شماره ۱: function annonymous

```
var x = function(){
  a > b  ?  x3 = ["a", a]  :  x3 = ["b", b];
   alert(x3) }

x( );
```

۶.۳ تابع عجیب شماره ۲: function invoked

```
var x = ( function(a, b){
  a>b ? x3=["a", a] : x3=["b", b] ;
   alert (x3) } ) (x1, x2)
```

۷.۳ تابع عجیب شماره ۳: closure

من اینجوری به خودم توضیح میدم: وجود "متغیر-پرانتز" بخاطر بازگشت تابع درونی بدون پرانتز

```
var x1 = 10;

function fn1(a){
   var b=16;
   function fn2(){return a/b}
   return fn2}

var x2 = fn1(x1)
```

البته من یجور دیگه هم این سه تابع عجیبو تحلیل میکنم. دوتای اولی توابعی هستن که توی یک متغیر ریخته شدن اما سومی چون اولش ریخته نشده آخرش ریخته شده!

3.8 BOM

BOM means Browser Object Model. yeah!! Browser is an object!

```
window.innerWidth //1280
window.open() //new-tab
      opens!
```

3.9 DOM-Properties

```
document.body
document.title
document.URL
```

DOM is a part of BOM. DOM is Document Object Model. DOM has properties like: body,title, url, ... It has methods like querySelector.

3.10 DOM-methods

```
document.getElementById('id1')
document.getElementByClassName('go')
document.querySelector('.class a , #id li a[href*="linkedin.com
    "]')
//stepo-1
document.createElement('figcaption')
//stepo-0
document.append(cpe)
```

3.11 DOM-Method.Method

```
//these make changes by themselves!!!
//common attributes
x6.setAttribute('style' , 'padding:1em; color:red; background-
    color: blue');
x6.setAttribute('id' , 'id1 id2 id3');
x6.setAttribute('class' , 'class1 class2 class3');
//attributed 4 form element and input element
x6.setAttribute('value' , 'submit');
x6.setAttribute('name' , 'mhmdamin');
x6.setAttribute('type' , 'text');
x6.setAttribute('type' , 'button');
x6.setAttribute('type' , 'radio');
x6.setAttribute('type' , 'checkbox');
x6.setAttribute('action' , 'index.html');
//img element has alt attribute!
x6.setAttribute('src' , '../img/001.png');
x6.setAttribute('alt' , 'a photo by mhmd amin khani');
//full-example
document.querySelector("#id3").setAttribute('style' , '
    background-color: blue; padding: 0 0 10px 0; border: solid
    black 2px');
//badihigat
var a1 = document.querySelector("#id3");
var a2 = document.querySelector("#id3 a");
var a3 = document.querySelector("#id3 a[href*='folder']");
//sometimes a2 is equal a3 but a1 is never aquals a2.
```

حالا اگه بخواهیم مقدار متن را عوض کنیم مثلا مقدار متنی را که داخل المنت p یا المنت input است انگاه بدین ترتیب عمل میکنیم:

```
var a4 = document.querySelector("p");
a4.innerHTML = "new text here!"
```

مرور کلی: بالایی ها برای attribute بودند برای داخل المنت هم از innerHTML و برای تغییر کل المنت از outerHTML استفاده میکنیم. مثال بخواهیم بجای تگ p تگ h1 باشد:

```
a4.outerHTML = "<h1>my title</h1>";
```

در پروژه ای که در بخش بعدی میبینید قصد داریم یک المنت بسازیم و قرار دهیم. خلاصه اینکه جاوااسکریپت به کد های html حتی یه ذره هم رحم نکرده! همه جوره داره تو کارشون دخالت میکنه! بیچاره! اگر همینطور پیش بره از کار بیکار میشه! خب وقتی همه کاراشو داره اسکریپت انجام میده دیگه چه دلیلی داره تو شرکت نگرش دارن؟!

3.12 Project

question

```
  //before starting project we have this codes in index.html
<figure>
  <img src="img/1.png" alt="this is photo about me">
```

```
</figure>
//after project we created new element in inspect-element. There
    is no change in index.html
<figure>
  <img src="img/1.png" alt=" ">
  <figcaption>this is photo about me</figcaption>
</figure>
```

solution

```
const FIGURE = document.querySelector("figure.go");
const IMG = FIGURE.querySelector("img");
var x = document.createElement("figcaption")
var x1 = IMG.getAttribute("alt")
x.append(x1)
FIGURE.append(x)
IMG.setAttribute("alt","_blank")
//main code doesnt change just the code in inspecth-element will
    be!
```

3.13 inline-style

```
var a = document.querySelector("div")
a.style.margin = "20px";
a.style.margin = "0 10px";
a.style.padding = "20px";
a.style.color = "green"; // just font-color changes!
a.style.fontFamily = "iran3"; //just <p> is effected not <h2>
a.style.backgroundColor = "red"; // hover goes off.
a.style.backgroundImage="url(001.png)";
a.style.cssText = "margin: 10px 0 ; border: solid black 2px";
```

شماره ۱۰۰: این راهو یادته؟ :-)

```
x.setAttribute("style","margin: 10px 0 ; border: solid black 2px
    ");
```

3.14 comparison-notes

```
document.querySelector("img[alt*='_b']")
document.querySelector("img[alt*='']")
document.querySelector("img").hasAttribute("alt")
```

۱۵.۳ خرده پروژه ۳: هنگام هاور رنگش عوض بشه!

وقتی موس میره روش رنگش عوض بشه!
هاور در استایل و آن موس آور - آن موس اوت در اسکریپت کار همدیگرو میکنن فقط اسمشون فرق داره!

```
var a = document.querySelector("#id3");
function fno(){a.classList.add("blu")}
function fn1000(){a.classList.remove("blu")}
a.onmouseover=fno;
a.onmouseout=fn1000;
```

۱۶.۳ خرده پروژه ۴: ساخت یک المنت ساده

ساخت یک المنت، ساده تر از قبلی!

```
var x2 = document.createElement("img");
x2.setAttribute("src","001.png");
var x1 = document.querySelector("#id2");
x1.append(x2);
```

۱۷.۳ نوزدهم مرداد ۹۸

کامل نوشتن برای بهتر یادگرفتن: تصمیم گرفتم از این به بعد همه کد های ممکن که به ذهنم میرسه یا پروژه هایی که به ذهنم میرسه رو بنویسم. اشتباه اگر قراره بکنم بهتره همین الان بکنم تا وسط پروژه های بزرگ!

واضح شدن یک تصویر بزرگتر: همچنین پازل که تکمیل میشود ارتباط تکه کد ها با هم مشخص میشود. حرفه ای ها هم به همین سبک پایه خود را قوی میکنند. با یک نگاه میتوان تمام حالت های ممکن را به خاطر سپرد! کامل نوشتن یعنی تکمیل یک پازل، یک تصویر بزرگتر!

```
var a = document.querySelector("#id1");
a.classList.toggle("hide");
a.classList.add("hide");
a.classList.remove("hide");
a.classList.contains("hide");
```

شماره ۲۰۰: با اینا میشه محتوا تغییر داد

```
a.innerHTML="...";
a.outerHTML="...";
a.innerText="hello!";
```

شماره ۲۰۱: مثال از بالاییا

نکته مهم اینه که بدونید میشه با یک دکمه F12 در بروزر کروم به console رفت و از انجا اینها را تست کرد و دید که اوتر یعنی چی و اینر یعنی چی. اوتر یعنی کل المنت همراه با خودش اما اینر یعنی تمام المنت های درونش! حالا میبینید چه راحت محتوا تغییر میکنه؟! :-)

```
a.innerHTML="<h1>hello</h1><img src="001.png" alt="hello">";
a.outerHTML="<div class="gold"><h1>hello</h1><img src="001.png"
    alt="hello"></div>";
```

پیام: میشه توی یه متغیر دیگه ریختشون و حملشون کرد به جای دیگه.

```
var b = a.innerHTML;
var b = a.outerHTML;
```

شماره ۲۰۲: ادامه متد های یک المنت

```
a.id
a.classList
a.classList.toggle("hide")
```

شماره ۲۰۳: میشه تو یه خط ای دی تغییر داد.

```
var a = document.querySelector("#id1");
a.id = "id2";
```

شماره ۲۰۴: یا اینکه ای دی رو توی یه متغیر ریخت.

```
var b = a.id;
var b = document.querySelector("#id1").id;
```

شماره ۲۰۵: یه عکس دسته جمعی :)

```
var a = document.querySelector("#id1");
a.innerHTML
a.outerHTML
a.id
a.classList
a.src
```

اون اخری برای المنت img نوشته شده! مگه فقط المنت img اتریبیوت src داره؟ مگه تگ script یا link این اتریبیوت src را ندارند؟! :-)

۱۸.۳ بیستم مرداد ۹۸

شماره ۲۰۶: مقدار گیر ها که میگن اینان! :-) بشناسیدشون! اینا کارشون همینه که فقط المنتو و اتریبیوت بگیرن! خیلی حرفه ای هم عمل میکنن! خیلی دقیق میتونی بهشون ادرس بدی که کیو بگیرن!

```
var x3 = document.querySelector("#id3");
var x4 = document.querySelector("#id3 a ");
var x5 = document.querySelector("#id3 a[href*='folder']");
var x6 = document.querySelector("#id3 a[href*='folder']").
    getAttribute("href");
```

شماره ۲۰۷: برای تعریف متغیر x6 اینطوری هم میشه

```
var x6 = x5.getAttribute("href");
```

شماره ۲۰۸: فکر کردی این خانواده نمیخواد بچه دار بشه :-)

```
var x7 = x5.innerHTML;
```

شماره ۲۰۹: تازه این بچه بلده با گریم خودشو به شکل دیگه ای هم نشون بده!

```
var x7 = x5.innerText;
```

شماره ۲۱۰: حالا این عکس ارزششو داره که یبار دیگه گرفته بشه!

```
var x3 = document.querySelector("#id3");
var x4 = document.querySelector("#id3 a ");
var x5 = document.querySelector("#id3 a[href*='folder']");
var x6 = document.querySelector("#id3 a[href*='folder']").
    getAttribute("href");
var x7 = document.querySelector("#id3 a[href*='folder']").
    innerHTML;
```

شماره ۲۱۱: با فامیلاشون میخوای اشنا بشی؟ :-)

```
var x41 = document.querySelector("form input[type*='button']");
var x42 = x41.value;
```

شماره ۲۱۲: این راه سخترشه، اخه بالایی اسون تره! زودتر گفتمش :-)

```
var x42 = x41.getAttribute("value");
```

شماره ۲۱۳: بابا بزرگشو ببین همه نوه هاشو چجوری دور خودش جمع میکنه!

```
var x4 = document.querySelectorAll("div");
```

۱۹.۳ خرده پروژه سه خطی من :-)

شماره ۲۱۴: ببینید چه خبره! پروژه فصل دو اما فقط در سه خط!
اینم واسه خودش کسی شده! رفته اتریبیوت Alt را از المنت img داخل id2 گرفته بعد همونو چسبوندتش به تگ p . نیگا چه قاراش میشی شده! آخرشم همرو ریخته توی متغیر بعد همون متغیرو باز چسبونده ته همون id2 ! آدم نمیدونه چی بگه والا! از دست اینا!

```
var x1 = document.querySelector("#id2 img").alt;
var x2 = document.createElement("p").append(x1);
document.querySelector("#id2").append(x2);
```

۲۰.۳ خرده پروژه ای که در برگه html نوشته شد!

شماره ۲۱۵: رفتم گذری زدم به دام! قطعه کد زیر در index.html نوشته شده است. نمیدونستم تگ باتن یک اتریبیوت داره که شبیه ایونت اسکریپت میمونه! منظورم onclick هست.

```
<button onclick="fn3()" >click here!</button>
      <script type="text/javascript">
        function fn3(){
          document.open();
          document.write("<h1>Welcome to Foundation</h1>");
          document.close();
        }
      </script>
```

۲۱.۳ خرده پروژه: بهت میگم چندبار کلیک کردی! :-)

```
<button onclick="fn3()">click here!</button>
      <script type="text/javascript">
        var x3=0;
        function fn3(){
          document.querySelector("button").onclick= x3++;
          document.querySelector("p").innerHTML="button is
              pressed "+x3+" times!";
        }
      </script>
```

۲۲.۳ خرده پروژه: مسابقه شش خطی

شماره ۲۱۸: دوباره رفتم پایه رو قوی کنم! حرفه ای ها این تفاوت هارو حس میکنن! در شش خط کد زیر هر شش متغیر تلاش کردند تا به یک المنت برسند. اما فقط دو تای انها موفق نشدند!

```
var x3 = document.querySelector("#id2 img #id10");//null
var x4 = document.querySelector("#id2 #id10");
var x5 = document.querySelector("div #id2 #id10");
var x6 = document.querySelector("div #id2 img #id10");//null
var x7 = document.querySelector("div #id2 img");//
var x8 = document.querySelector("#id2 img");//
var x9 = document.querySelector("div #id10");//
```

۲۳.۳ تفاوت را احساس کنید!

تفاوت را احساس کنید! فقط یک space فرقشونه!

```
var x110 = document.querySelector("#id2 p .class1");//null
var x110 = document.querySelector("#id2 p.class1");
```

۲۴.۳ خرده پروژه: کلاس های یک دیو را به دیو دیگر دادن

```
var x80 = document.querySelector("#id2").classList;
document.querySelector("#id1").classList = x80 ;
```

۱.۲۴.۳ نکته: فقط خط اول درسته!

```
1 document.querySelector("#id1").classList.add("class1") //just
      one class
2 document.querySelector("#id1").classList.add("class1 class2
      class3") // false
3 document.querySelector("#id1").classList.add("class1,gold,class3
      ") // false
```

فصل ۴

ربع سوم: رویدادها events

۱.۴ خرده پروژه: دکمه و متن پنهان شونده

دکمه رو فشار بدی متن میاد میره :-)

```
var x = document.getElementById("id1");
var y = document.getElementById("id2");
function fn1(){
  y.classList.toggle("hide");
}
x.onclick = fn1;
```

میخوای متن روی دکمه هم عوض شه! اینو به تابع اضافه کن!

```
  x.innerHTML == "see!" ? x.innerHTML = "hide!" : x.innerHTML =
      "see!" ;
```

راه سخترش اینه که یک تابع دیگه تعریف کنی! اونوقت addEventListener لازم میشه! تا هر دو تابع با یک کلیک همزمان اجرا بشن!

```
x.addEventListener("click",fn1, false);
x.addEventListener("click",fn2, false);
```

ممکنه از else if بخواهید استفاده کنید!

```
if (x.innerHTML == "see!") {
    x.innerHTML = "hide!"
  } else {
    x.innerHTML = "see!"
  }
```

ساده تر هم میتونه بشه! ببینید! ولی کار نمیکنه! :-) میدونی چرا ...

```
var x50 = x.innerHTML;
var x51 = "see!";
var x52 = "hide!";
  if (x50 == x51) {
    x50 = x52;
  } else {
    x50 = x51;
  }
```

چون آخرش باید بریزیم توش!

```
x.innerHTML  =  x50;
```

۲.۴ کمی از جیسون JSON

اینجا یک ابجکت تعریف کردیم و خواستیم نمایش بدیم. چون بروزر نمیتوانست ابجکت را نشان دهد از جیسون استفاده کردیم. همانطور که میبینید کنسول لوگ بدون جیسون هم میتواند اشیا را نشان دهد.

```
var gold = {
  title: "course",
  author: "me"
}
document.querySelector(".foot").innerHTML = JSON.stringify(gold)
    ;
console.log(gold);
```

موارد استفاده از جیسون: برای نمایش یک ابجکت داخل یک دیو بصورت html یا alert دادن یا فرستادن برای لاگین شدن از جیسون استفاده میشه! مثلا تگ فرم که متد پست داره!

یه لحظه قاطی کردم :-) در اوایل فکر میکردم اینجوریه!

```
document.querySelector(".foot").innerHTML = gold.JSON.stringify
    ;
```

بازگشت به ابجکت: یکی استرینگ میکنه و دیگری به حالت اول برمیگردونه!

```
let x = JSON.stringify(xhr);
let y = JSON.parse(x);
```

الان اینجا y همون xhr هست. y یک ابجکت هست.

اطلاعات بیشتر: http://javascript.info/json#json-stringify

۳.۴ کمی از Ajax

اینو توی بروزر هم میشه دید

```
let xhr = new XMLHttpRequest();
console.log(xhr);
```

همینطور ایشون هم در بروزر خودشونو نشون میدن

```
var x1 = document.querySelector("#btn1")
x1.addEventListener('click',math);

function math() {
  var xhr = new XMLHttpRequest();
      console.log(xhr)
  }
```

اما ایشون کلاسشون به لوکال هاست میخوره!

```
var x1 = document.querySelector("#btn1")
x1.addEventListener('click',math);

function math() {
  var xhr = new XMLHttpRequest();
  xhr.open('GET','1.txt',true)
  xhr.onload = function (){
    if (this.status == 200) {
      console.log(this.responseText);
    }
  }
  xhr.send();
  }
```

منم که عاشق لوکال هاست زمپ :-)

فصل ۵

ربع آخر: حلقه، تروبلشوتینگ

هنوز چیز زیادی در حد یک دولوپر از اسکریپت یاد نگرفتم اما تلاشم قابل تقدیره!

۱.۵ یه حلقه ساده

```
for(var i=0; i < 10; i++){
  document.querySelector("#id2 p").append(i)
} // output: 123456789
```

۲.۵ یک حلقه ساده دیگر

ابتدا a ها را میگیرد و یک ارایه میسازد. a.length تعداد این تگ ها را نشان میدهد. وقتی از حلقه بخواهیم که به تعداد تگ ها i را تولید کند انچه اتفاق می افتد این است که به تعداد تگ ها دقیقا به همان تعداد حلقه تکرار میشود و هر دستوری در حلقه تعریف کنیم برای هر یک از تگ ها اتفاق می افتد. برای اینکه همه تگ ها را احضار کنیم کافیست بنویسیم x[i]

```
var x10 = document.querySelectorAll("a");
for(var i = 0; i < x10.length; i++){
  var x11 = x10[i].setAttribute("target","_blank")}
```

الان x11 ارایه ای هست که اعضایش تگ های a هستند اگر بخواهم این تگ ها را نشان دهم کافیست یک حلقه دیگر برای نمایش اعضای این ارایه، تعریف کنم. نه! اشتباه گفتم! :-) حلقه هم تست کردم جواب نداد چون کد است و برای نمایش کد در میان کد باید از prisma استفاده کرد. همان کاری که برای تهیه epub برای بوتسترپ انجام دادم :) گاهی حرفه ای هستما ! ماشالا!
https://prismjs.com/

۳.۵ مقدمه: مقدار های src را بگیرید!

```
var x100 = document.querySelectorAll("#id2 img");
for(var i=0; i < x100.length; i++){
  var x101 = x100[i].getAttribute("src");
  document.querySelector("#id2 p").append(x101);
} //output: 001.png001.png001.png
```

تازه بهتر از بالا جایگذاری اینه:

```
  var x101 = x100[i].getAttribute("src").slice(0,-4);
```

که در نهایت خروجیتون میشه این:

```
 // output: 001001001
```

حالا اگه بخوایم هر خروجی در یک سطر نمایش داده بشه:

```
var x100 = document.querySelectorAll("#id2 img");
for(var i = 0; i < x100.length; i++){
  var x101 = x100[i].getAttribute("src").slice(0,-4);
  var x102 = document.createElement("br");
  document.querySelector("#id2").append(x102);
  document.querySelector("#id2").append(x101);}
```

قبل از اینکه کدهای بالارو بنویسم این قطعه کد اشتباه را نوشته بودم. بنظرتان مشکل کجاست؟

```
var x100 = document.querySelectorAll("#id2 img");
for(var i = 0; i < x100.length; i++){
  var x101 = x100[i].getAttribute("src").slice(0,-4);
  var x102 = document.createElement("p").append(x101);
  document.querySelector("#id2").append(x102)}
```

ایده ای بذهنم رسید: من چند روز پیش میخواستم با کلیک بره عکس بعد ولی نمیدونستم چجوری حالا فهمیدم با slice میشه jpg. حذف کرد تا بشه یکی به عددی که در قسمت نام امده اضافه کرد دوباره jpg. را به ان داد و به src برگرداند. حالا کلیک کنی عکس بعدی از داخل پوشه نشون میده! اما در وب اینطور نیست! در پروژه ویندوزی ما که اینجا برای تست درستش کردیم جواب میده! برای وب از ajax شاید باید استفاده کرد. یادمه برای فراخوانی ویترین از PDO در زبان برنامه نویسی php استفاده میکردیم. از حلقه هم استفاده میکردیم. و همچنین از SELECT استفاده میکردیم. یادته محمد امین؟

۴.۵ آخرین بیگ پروژه این کتاب: Responsive-Images-Attributes

۱.۴.۵ میخوایم چیکار کنیم؟

در این پروژه میخواهیم تمام تصاویر را (اینجا شش تصویر داریم) رسپانسیو کنیم. رسپانسیو کردن با مقدار دهی به دو تا اتریبیوت جدید المنت img (srcset ، sizes) همین! کل پروژه همینه!

۲.۴.۵ پیش نیاز => تعریف دو اتریبیوت ،sizes srcset

In HTML5 two new attributes is defined for img tag for responsive images. ”sizes” & ”srcset”

sizes A list of one or more strings separated by commas indicating a set of source sizes. Each source size consists of: a media condition. This must be omitted for the last item. a source size value.

srcset A list of one or more strings separated by commas indicating a set of possible image sources for the user agent to use. Each string is composed of: a URL to an image, optionally, whitespace followed by one of: A width descriptor, or a positive integer directly followed by ’w’. The width descriptor is divided by the source size given in the sizes attribute to calculate the effective pixel density. A pixel density descriptor, which is a positive floating point number directly followed by ’x’.

Source: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img#Attributes

۳.۴.۵ پیش نیاز => انچه در index.html است:

```
<div>
  <p>this is a paragraph</p>
  <p class="class1"> </p>
  <img src="img/001-720px.jpg" data-type="paper">
  <img src="img/001khat-720px.jpg" data-type="paper">
  <img src="img/natures-720px.jpg" data-type="nature">
  <img src="img/shaki-720px.jpg" data-type="nature">
</div>

<img src="img/cosmos-720px.jpg" data-type="flower">
<img src="img/rose-720px.jpg" data-type="flower">
<img src="img/natural-720px.jpg" data-type="flower">
```

۴.۴.۵ پیش نیاز => لیست تصاویر فولدر img :

در برگه ۷ تا تگ img داریم درحالیکه در پوشه ۲۱ عکس داریم. زیرا از هرعکس ۳ سایز برای رسپانسیو شدن داریم. قرار است srcset ، sizes بین این سه سایز یکی را انتخاب کنند تا به تناسب عرض صفحه نمایش با توجه به تعریفی که در ادامه به انها میدهیم تصاویر را انتخاب کنند.

```
001-720px.jpg
001-600px.jpg
001-480px.jpg
001khat-720px.jpg
001khat-600px.jpg
001khat-480px.jpg
natures-720px.jpg
natures-600px.jpg
natures-480px.jpg
shaki-720px.jpg
shaki-720px.jpg
shaki-480px.jpg
cosmos-720px.jpg
cosmos-600px.jpg
cosmos-480px.jpg
rose-720px.jpg
rose-600px.jpg
rose-480px.jpg
natural-720px.jpg
natural-600px.jpg
natural-480px.jpg
```

هر چند عکس یک data-type یکسان دارند. در مجموع ۳ عدد دیتا-تایپ داریم.

۵.۴.۵ مرحله ۰: گرفتن تمام المنت های img با querySelectorAll

```
const IMAGES = document.querySelectorAll("img");
```

۶.۴.۵ مرحله ۱: گرفتن همه src هایش و data-type ها فقط با یک حلقه

در مرحله اول یک حلقه تعریف کردیم. در این حلقه فقط src ها را گرفتیم. (مثل مقدمه بالا)

```
for (let i = 0; i < IMAGES.length; i++) {
  let imgSrc = IMAGES[i].getAttribute("src").slice(0,-10);
  console.log(imgSrc);

  let type = IMAGES[i].getAttribute("data-type");
  console.log(type);
}
```

۷.۴.۵ مرحله ۲: ساخت تابع makeSrcset

در مرحله دوم تابع makeSrcset را راه انداختیم بعد در حلقه فراخوانی کردیمش!

```
function makeSrcset(imgSrc){
let width = 480;
let x = [];
for (var i = 0; i < 3; i++) {
  x[i] = imgSrc + "-"+ width+"px.jpg "+width + "w";
  width += 120;
}  return x.join()
}
```

این خط کد به حلقه اضافه میشود.

```
let srcset = makeSrcset(imgSrc);
```

۸.۴.۵ مرحله ۳: ساخت ابجکت جهت طبقه بندی دیتاتایپ ها

در مرحله سوم باز کمی حلقه را دستکاری کردیم بطوریکه اول ابجکت تعریف کردیم که سه تا data-type ها را در ان تعریف شد .

```
const Y = {
  nature: "100vw",
  flower: "(max-width: 799px) 100vw, 372px",
  paper: "(max-width: 799px) 100vw, 558px"
}
```

این خط کد به حلقه اضافه میشود.

```
let sizes = Y[type];
```

۹.۴.۵ مرحله ۴: برگرداندن مقدارهای به دست امده به صاحبانش

در پایان هم با setAttribute دو اتریبیوت جدید img را به ان اضافه میکنیم. در نهایت انچه در برگه اسکریپت خواهید دید این است:

```
const IMAGES = document.querySelectorAll("img");
const Y = {
  nature: "100vw",
  flower: "(max-width: 799px) 100vw, 372px",
  paper: "(max-width: 799px) 100vw, 558px"
}

function makeSrcset(imgSrc){
let width = 480;
let x = [];
for (var i = 0; i < 3; i++) {
  x[i] = imgSrc + "-"+ width+"px.jpg "+width + "w";
  width += 120;
}  return x.join()
}

for (let i = 0; i < IMAGES.length; i++) {
  let imgSrc = IMAGES[i].getAttribute("src").slice(0,-10);
  let srcset = makeSrcset(imgSrc);
  IMAGES[i].setAttribute("srcset", srcset);

  let type = IMAGES[i].getAttribute("data-type");
  let sizes = Y[type];
  IMAGES[i].setAttribute("sizes", sizes);
}
```

۵.۵ ارور ها

راه ۱: برای کنترل ارورها کافیه بری به کنسول لوگ خودش میگه کدوم خط ارور داره!

راه ۲: میتونی یک پلاگین به نام Eslint هم در atom نصب کنی که بگه! البته بهتره تنظیماتشو بذاری روی بک گراند کار کنه!

راه ۳: میتونی کدهایت را به سایت زیر کپی پیست کنی فقط نکتش اینه که در تنظیماتش es۶ فعال کنی!

https://jshint.com

نکته: در نهایت برای بهبود عملکرد میتونید از سایت زیر استفاده کنید!

http://minifier.org

فصل ۶

کامپایل کردن ES۶ به ES۵

برای کامپایل compile یا transpiling یک زبان برنامه نویسی از سطحی به سطح دیگر که برای بروزر خوانا شود کافیست از بابل یا وبپک استفاده کنید! در ادامه به نصب و کاربرد های اولیه این دو میپردازیم.

بروزر chrome-canary را نصب میکنیم که طلایی رنگ است. در فصل اول Rjs-latex به اکما اسکریپت اشاره کردیم. (کتابچه بعدی می باشد!) اکما اسکریپت ۶ برای بروزر خوانا نیست. شاید دلیل نامگذاری اش سال ۲۰۱۶ باشد. اکما اسکریپت ۷ هم داریم. اکما اسکریپت ۶ همان اکما اسکریپت ۵ است اما با قواعد نوشتاری به روزرسانی شده! بابل و وبپک، دو پکیج از محیط node هستند که این تبدیل را انجام میدهند. همچنین بصورت انلاین در سایت بابل یا سایت های دیگر و یا با خود محیط nodejs هم میتوان این تبدیل را انجام داد.

آنچه در این فصل میخوانید:

- Inbrowser Babel transpiling
- Transpiling ES6 with Babel and babelnode
- Transpiling with Webpack

۱.۶ نصب کردن babel

ابتدا به nodejs.org بروید و node نصب کنید تا terminal ساخته بشه! قبل از نصب هر پکیج در محیط node بایستی فایل package.json ساخته شود!

npm init

خط به خط سوال میپرسد تا فایل جیسون را پر کند. پس از اتمام ان:

npm install -g babel npm install -g babel-cli

اگه اینو بزنی هیچی نیاد یعنی درست نصب شده!

babel

۲.۶ استفاده از babel

حالا script.js را که در مسیر زیر ساختم و درونش کمی کد اسکریپت است را میخواهم با بابل کامپایل یا همان transpiling کنم.

drive c > users > shervin > script.js

بدین منظور هرکدام از کد های زیر میتواند راهگشا باشد:

babel script.js

babel script.js –out-file script-compiled.js

اگر از دومی استفاده کنیم یک فایل جدید میسازد و کد های کامپایل شده را در ان میریزد!

Clear

babel script.js –watch –out-file script-compiled.js

اگر بخوایم ببینیم درست کامپایل شده به es۵ کافیه از سایت زیر استفاده کنیم:

`https://ES6console.com/`

۳.۶ استفاده از babel-node

بریم babel-node استفاده کنیم برای تبدیل!

babel-node script.js

البته اینم میشد

node script.js

این ران میکنه پروژه رو نه اینکه بازکدنویسی کنه

پای متن:

برای نوشتن همین متن علاوه بر تماشای lynda-ES۶-upandRunning (قابل دانلود از لینک زیر) به سه لینک دیگر هم سر زدم و درنهایت پس از دو روز نصب و حذف نصب و تمرین توانستم نحوه نصب بابل را یاد بگیرم. هنوز هم بصورت گلوبال نه لوکال! ولی کارمون راه افتاد!

https://babeljs.io/en/setup
http://babeljs.io/docs/usage/cli/
https://www.npmjs.com/package/@babel/core
http://navid.kashani.ir/520/package-json/

نکته:

تا الان بروزر من بدون بابل هم ES۶ را میخواند. پس متن ES۶ چه شکلیه؟ به لینک زیر رفتم.
https://dev.to/srebalaji/ES6-for-beginners-with-example-c7
و کد های زیر را دیدم

```
// Old Syntax
function oldOne() {
 console.log("Hello World..!");
}
// New Syntax
var newOne = () => {
 console.log("Hello World..!");
}
```

فصل ۷

قاعده نوشتاری ES۶-Syntax

۱.۷ استرینگ جدید مبارک! template-string

دو خط زیرین هیچ تفاوتی با هم ندارند. ولی خط اخر ES۶ هست و بهش میگن back-qoutation یا همون back-qout

```
var x2 = "mohammad"
var x3 = "<p>hello everyone! hello" + x2 + "!</p>";
var x4 = `<p>hello everyone! hello ${x2}! </p>`;
```

در کدهای پایین فقط خط اول درست اجرا میشود! (بدون ارور)

```
var x5 ="<pre>hello everyone!\nhello "+ x2 +"!</pre>";
var x5 ="<pre>hello everyone!
hello "+ x2 +"!</pre>";
```

همچنین میتوان برای نشان دادن در خط جدید از کد زیر هم استفاده کرد! (علاوه بر خط اول کدهای بالایی)

```
var x6 = `<pre>hello everyone!
hello ${x2}!</pre>`;
```

اگر تگ pre نداشته باشد در دو خط نشان نمیدهد!

۲.۷ ارایه در ارایه با سه خط spread-operators

در خط دوم ES۶ رعایت شده و اگر در کنسول لوگ ببینید چهار عضو میبینید!

```
let gol = ["go",10]
let gold = [0,"mhmd", ...gol]
let gold = [0,"mhmd", gol]
```

۳.۷ تابع ارایه ای

در ES۶ یک متغیر را میتوان برابر تابع => قرار داد. هنگام فراخوانی متغیر با پرانتز که توی پرانتز ارایه بدیم بجای یک عدد یا استینگ انگاه ارایه وارد تابع میشود. جالبه مگه نه؟

```
var x = (students) => console.log(students);
x(["mhmd","ali","hssin"]);
```

اگه توجه کرده باشید به تابع نام ندادیم. اسمش همون اسم متغیری هست که ریختیم توش تابعو!

اگر بخوایم فرق این تابع جدید را با سبک قدیمی ان مقایسه کنیم دو تا چیزش کم شده بجاش دو علامت اضافه شده. یکی اون } و دیگری کلمه function که بجاش => جایگزین شده! ولی باکلاس شده ها :-)

اینطوری هم میشه نوشت ولی قدیمیه!

```
var x = function (students) {console.log(students);}
x(["mhmd","ali","hssin"]);
```

اینجا هم اسمی به تابع ندادیم و از اسم متغیری که تابع توش ریخته شده بود استفاده کردیم تا تابع را فراخوانی کنیم.

۴.۷ گستره this

این شکل قدیمی تابع درش بکار رفته

```
var gold = {
  title: "mhmd",
  scores: [20,30,40,45,50,60],
  printScores: function () {
    var x = this;
    this.scores.forEach(function (score) {
      let y = x.title +" likes to "+score ;
      console.log(y)}) }}
gold.printScores();
```

اگه نخوایم this را بریزیم به x کافیه بعد تابع از متد bind() استفاده کنیم

```
...
this.scores.forEach(function (score) {
  let y = this.title +" likes to "+score ;
  console.log(y)}.bind(this))
...
```

این شکل جدید تابع درش بکار رفته همچنین printScores پرانتز گرفته!

```
...
printScores() {
  this.scores.forEach(score => {
    let y = this.title + " likes to " + score ;
    console.log(y)}
    )}
...
```

۵.۷ خراب کردن وظیفه

۱.۵.۷ بخش ۱ <= آرایه

```
var [x1,x2] = [200,201,202]
console.log(x1) //200
console.log(x2) //201
```

در مثال زیر میخواستیم سومین عضو از ارایه را بگیریم. به همین دلیل فاصله انداختیم.

```
var [x1,,x2] = [200,201,202]
console.log(x2) //202
```

۲.۵.۷ بخش ۲ <= ابجکت

```
var {title , why} = {
  title: "mhmd",
  go: "danmark",
  why: "vacation"}
console.log(why) //vacation
```

برای ارایه ها میشه اسم دیگری گذاشت ولی برای ابجکت ها حتما باید همان اسم داخل خودش باشد. مثلا این کار نمیکنه!

```
var {x1, x2} = {
  title: "mhmd",
  go: "danmark",
  why: "vacation" }
```

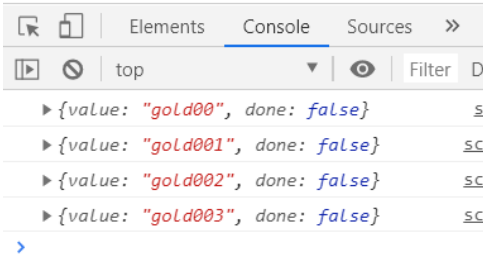
۶.۷ بیگ پروژه: یک تابع که نمیدونم چطور بگم چیکار میکنه :-)

```
var x = function ({happyPlace,date}) {
  return `I am going to go to the ${happyPlace} on ${date}`
}
var x1 = {
  happyPlace: "Top of the Letmalcan Mountains",
  necessaryThings: "Mineral Water and Some Cookies",
  date: "this Monday"
}
console.log(x(x1));
```

۷.۷ خرده پروژه: ژنراتور ها

```
function* fn1() {
   yield "gold00";
   yield "gold001";
   yield "gold002";
   yield "gold003"}
var x = fn1();
console.log(x.next());
console.log(x.next());
console.log(x.next());
console.log(x.next());
```

در خروجی این تصویر دیده میشود



حالا میخواهیم فقط یک متن نمایش داده شود نه ابجکت! کافیست یک value به اخر ان اضافه کنیم:

```
console.log(x.next().value);
```

۸.۷ بیگ پروژه: ژنراتور

۱.۸.۷ فاز اول پروژه: یه ژنراتور با حلقه تعریف کن!

```
function* fn1(x){for(let i = 0; i < x.length; i++){yield x[i]}}
var x1 = fn1([10,20,30]);
console.log(x1.next().value);
console.log(x1.next().value);
console.log(x1.next().value);
```

در خروجی این تابع خواهیم دید اعداد ده، بیست و سی در سه خط نمایش داده میشود.

۲.۸.۷ فاز دوم پروژه: فقط یه جمله به اخرش بچسبون!

حال همین را با setInterval که فعلا خودمم نمیدونم چیکار میکنه قراره به کار ببریم. هدف از این قطعه کدهایی که الان اضافه میکنیم، اضافه کردن یک استرینگ به پایان ده بیست سی .

```
var ten20 = setInterval(fn2(),500)
function fn2() {var x1object = x1fn1.next();
  if (x1object.done) { clearInterval(ten20)
    console.log("horraaa... it successfully passed!")
  } else {console.log(x1object.value) }}
```

فقط مشکلش اینه که عدد اولو نشون میده اما اگه تابع را اسم ندهیم و بدون فراخوانی بنویسیم همه چیز درست میشود.

```
...
var ten20 = setInterval(function() {var x1object = x1fn1.next();
...
```

اینطور که برمیاد setInterval یجورایی نقش یه حلقه رو بازی میکنه! وگرنه چطور میشه همش نمایش داده بشه! اخه اگه یادتون باشه در مثال اول سه بار کنسول لوگ را نوشتیم. درسته؟

فصل ۸

کلاس های ES۶

آنچه در این فصل میخوانید:

- قاعده نوشتاری کلاس های ES۶
- وراثت در کلاس ها
- بوسیله کلاس ES۶ میتوانید کمپوننت های reactjs بسازید!
- کنترل state با ES۶ و reactjs

۱.۸ قاعده نوشتاری و وراثت در کلاس های ES۶

تابع حداقل یه پرانتز داشت خدا پدرشو بیامرز این کلاس که این چیزارو نداره

```
class x {constructor(x1,x2){this.x1 = x1;this.x2 = x2}
    fn1(){console.log(`my bike has ${this.x1} wheels
            in ${this.x2} color`)}}
 new x(2,"white").fn1();
```

این کلاس ES۶ دو قطعه دارد: تابع و کنستراکتور. تابع برای نمایش و کنستراکتور برای ابجکت کردن کلاس.
آخرشم برای اجرا مثل تعریف ابجکت جدید باهاش رفتار میکنیم. بهش new میدیم.

داستان یک ارور: دفعه اول که اجرا میکنم معمولا یه چیزو یادم میره! اینبار سر this.x۱ ارور داد :-)

۱.۱.۸ وراثت در کلاس ها

وراثت در کلاس با extends شروع میشه که بین کلاس جدید و قدیم استفاده میشه. بدیهیه که class اولش باید بذاریم.

```
class x100 extends x {
  constructor(){
    super("big bike",18)}}
```

به همین سادگی تعریف شد یعنی به وراثت رسید حالا کافیه اجراش کنیم تا در بروزر نمایش داده بشه

```
new x100().fn1()
```

۲.۸ بیگ پروژه: اولین کمپوننت من!

۱.۲.۸ پیش نیاز <= inbrowser-react

از اینجا به بعد که قراره react بیاد وسط لازمه که به سایت زیر بریم و اخرین خط را به inbrowser-script کپی کنیم.

https://cdnjs.com/libraries/react/0.14.0

این اسکریپت که در خط زیر در بالای تگ title در index.html قرار دارد.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/react
    /0.14.0/react.min.js"></script>
```

۲.۲.۸ پیش نیاز => inbrowser-babel

بعدشم باید بابل بیاد مثل همین ری اکت که اوردیم:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core
    /5.8.29/browser.js"></script>
```

کدها داخل برگه html نوشته شوند. اسکریپت از نوع بابل باشد.

```
<script type="text/babel">
```

۳.۲.۸ فاز اول پروژه: ساخت کلاس از طریق React

از یک رستوران میخوایم حرف بزنیم بنابراین یک متغیر تعریف میکنیم.

```
var Restaurant = React.createClass(
  { render: function(){
    return (<div>
      <h1>Hello World!</h1>
            </div>)}})
```

تا اینجا ما یک کمپوننت react component تعریف کردیم.

مرحله اجرا: حالا کمپوننت را فراخوانی میکنیم و مقدار دهی میکنیم:

```
React.render(<Restaurant name="goal"></Restaurant>
 , document.querySelector("div.class1"))
```

اگه بخوام بگم چه اتفاقی اینجا افتاده، یک رندر هست که پرانتز داره و داخل پرانتز دو مقدار با پرانتز از هم جدا شدن. دومیش میگه کجا نمایش داده بشه و در اولیش همون متغیری که در ابتدا با نام رستوران تعریفش کردیم شبیه تگ html نوشته شده! چه جالبه تعریف کردن یک کمپوننت در ری اکت! مگه نه؟

البته اولش که خروجی گرفتم مشکل داشت. میگفت رستوران تعریف نشده! راست میگفت بیچاره! من یک حرف را اشتباه نوشته بودم.

خیلی سعی کردم از inbrowser به فایل دیگری انتقال دهم کد هایم را اما نشد. قطعا راهی دارد.

کمپوننتی که از کلاس به ارث رسیده: اینجوری هم اگه بنویسیم درسته!

```
class Restaurant extends React.Component{
  render(){ return (<div> ...
```

۴.۲.۸ فاز دوم پروژه: state باید برقصه!

بخش اول: به قبل از تابع ()render دو تا تابع دیگه هم اضافه شدن
یکی همون constructor که توی هر کلاسی در ES۶ مهمون ماست
یکی هم ()openRestaurant که خودمون نیازش داریم تا دکممون کار کنه!

```
constructor(){
  super()
  this.state = {status: "not open"}
}
openRestaurant(){   this.setState({status: "open"});   }
```

در اینجا constructor یک بچه با خودش اورده که من اولین بارمه میبینمش ()super. همچنین اقای constructor در کنار super یدونه state هم داره که مثل پروپرتی ای که با یه نقطه به ابجکتش وصل میشه اومده به همین کلاس وصل شده. ولی بجای اسم بردن مستقیم از نام کلاس از this بهره برده! شاید بخاطر اینکه کلاس هم یجور ابجکته دیگه :-)

بخش دوم: یک پارگراف و یک دکمه اضافه کنیم به بعد از تگ h1
بخاطر اینه که ما یه دست راستی داشته باشیم که باهاش با بروزر ارتباط interaction برقرار کنیم.

```
<p>the Restaurant is {this.state.status}.</p>
<button onClick={this.openRestaurant}>open it</button>
```

بخش سوم: بهconstructor بایستی append کنیم

```
this.openRestaurant = this.openRestaurant.bind(this)
```

دیگه نمیگم چرا بدیهیه دیگه نه؟ ظاهرا this در تابع openRestaurant نمیتونسته به کلاس ارجاع بده!

پاراگراف زیریو همینجوری نوشتمش! میتونی نخونیش و یکراست بپری به بخش ۴ :)

میدونی چی شده بود برای من؟ من دکمه رو اینجوری نوشته بودم :) از دست تو محمد امین!

```
<button onclick="this.openRestaurant">open it</button>
```

تا حدودی هم میشه گفت حق داشتم بابا! من بیچاره یاد متد onclick توی دنیای اسکریپت افتاده بودم از طرفی هم جدیدا با این اتریبیوت اشنا شده بودم و این اتریبیوت همیشه مقدار خودشو داخل استرینگ میریخت منم فکر کردم اینبار هم استرینگ و دبل کوتیشن میخواد! نگو متغیر بوده و باید داخل علامت خاص خودش قرار داده میشده اگر در برگه html یه دکمه داشتیم که اتریبیوت onClick داشت و جلوش دبل کوتیشن داشت، بخاطر اینکه اون جای تابع بوده. مثل این:

```
<button onClick="fn1()">open it</button>
```

ایندفه باید بیشتر دقت کنم. دیگه یاد گرفتمش! ادم یه بار که اشتباه میکنه توی ذهنش ملکه میشه! نه؟

اگرم میخوای خودت تست کنی:

```
<button onClick="fn100()">hello </button>

<script type="text/javascript">
function fn100() {
  alert("gogogo")
}
</script>
```

بخش چهارم: دیگه اجازه بده من کداشو ننویسم! خودت بنویس یادبگیری!

میخوایم رستورانو ببندیم. عین قبلیه! فقط یک دکمه و یه تابع میخواد که هردوتاش قابل کپی کردنه! در دکمه در onClick بجای تابع قبلی تابع جدیدو مینویسیم. در تابع هم یه اسمش باید عوض بشه و مقدار status هم تغییر کنه به not-open . یدونه هم bind(this) میخواد! همین!

```
class Restaurant extends React.Component{
  constructor(){
    super();
    this.state = {status: "not open"};
    this.openRestaurant = this.openRestaurant.bind(this);
    this.closeRestaurant = this.closeRestaurant.bind(this);
  }
  openRestaurant(){  this.setState({status: "open"});  }
  closeRestaurant(){  this.setState({status: "not open"});  }
  render(){return (<div>
    <h1>{this.props.name}</h1>
    <p>the Restaurant is {this.state.status}.</p>
    <button onClick={this.openRestaurant}>open it</button>
    <button onClick={this.closeRestaurant}>close it</button>
    </div>)}}

React.render(<Restaurant name="goal"></Restaurant>, document.querySelector("div.class1"))
```

قصه ما به سر رسید! :-) کلاغه به خونش نرسید! بله محمد امین جان! دنیا روز های خوش هم داره! الانم یکی از اون روزاست! این فصل که اخرین فصل بود به اتمام رسید! نگارش این کتاب هم به اخرین جملاتش نزدیک شد!

خوشحالی ام را نمیدانم چگونه وصف کنم. در پوستم نمیگنجم! دوستتون دارم خیلی زیاد! کمپوننت های عزیز قرار است در اینده بیشتر از شما سر دربیاورم! به امید دیدار در کتاب بعدی!

فصل ۹

به سرزمین Reactjs رسیدیم!

آنچه در این فصل خواهید خواند:

- نصب دو افزونه کروم
- لینک های CDN برای ReactDOM
- اولین React من :)
- نصب پکیج httpster

۱.۰.۹ افسانه دو برادر

دو افزونه گوگل کروم به نام react-detector و دیگری به نام React-developer-tool نصب کنید! حالا به سایت reactjs.org بروید و دکمه F۱۲ را بزنید! در کنار console ، elements و ... یک گزینه اضافه شده که اگه برید توش کمپونت هایی که با ری اکت ساخته شدنو نشونتون میده!

۱.۹ اولین React من :)

۱.۱.۹ پیش نیاز (۱ گام)

۱.۱.۱.۹ اماده کردن برگه html

لینک CDN : به سایت reactjs.com رفتم و این لینک هارو برای product گیر اوردم. مثل همیشه به بالای title افزودم.

https://reactjs.org/docs/cdn-links.html

گام ۱: تگ زیر append شود به تگ body !

```
<script src="index.js">
</script>
```

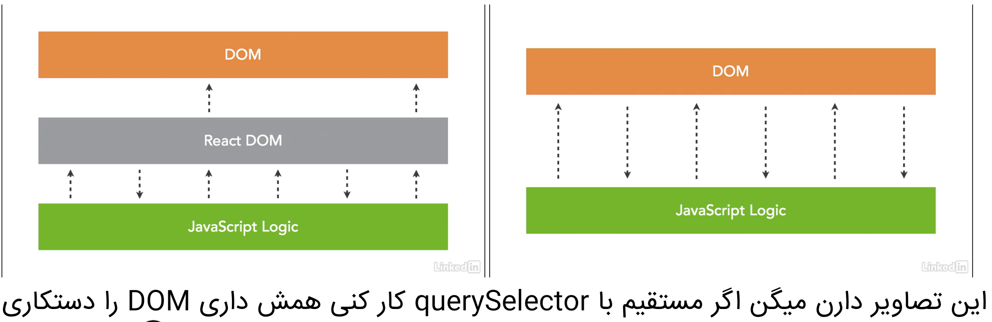
گام ۲: این تگ را نوشتم:

```
<div id="react-container">
</div>
```

۲.۱.۹ اینم از اولین کد من!!

خانه اولین کد react من در برگه index.js می باشد:

```
const title = React.createElement('h1',
{id: 'title', className: 'header'},
'Mohammad ')
ReactDOM.render(title,
document.querySelector("#react-container"))
```



این تصاویر دارن میگن اگر مستقیم با querySelector کار کنی همش داری DOM را دستکاری میکنی اما همیشه دستکاری نیاز نیست! بخاطر همین ReactDOM میاد وسط! 😮

۱.۲.۱.۹ چجوری حفظش کنم؟

این اسکلتی که مشاهده میکنید داره درباره تعداد صندلی هایی که رزرو شدند صحبت میکنه!

```
const title = React.createElement(--,--,--)
ReactDOM.render(--,--)
```

۳.۱.۹ تست بروزر (۲ گام)

۱.۳.۱.۹ راه اندازی هاست httpster

پکیجی از نود هست به نام httpster که یک لوکال هاست هست که رقیب زمپ هست و قوی تره بنظرم. فعلا نیازی به روشن کردن http از طریق پکیج httpster نیست. اما اگر میخواهید افسانه دو برادر در بروزر فعال شوند، هاست راه بندازید!

۱. به ترمینال رفتم و کد npm زیر را وارد نمودم:

npm install -g httpster

(میدونم اگه نمیگفتمم خودت بلد بودی!)

۲. ادرس ترمینال را دادم:

cd Desktop/Rjs-html

۳. پروتکل آتیش کردم: (هر عددی که دلتان میخواهد بگذارید!)

httpster -p 4000

۲.۳.۱.۹ مشاهده در browser

حالا بروزر باز کنید در ادرسش بزنید:

localhost:4000

انگاه بالا میاره صفحه تان را و شادتان میکند :)
بعد هر دفعه هم که تغییراتی میدیم یادمه در لیندا این هاستو اپدیت میکرد. برای اپدیت میرفت به ترمینال ctrl+c میزد تا دوباره http روی پروتوکل اجرا بشه! تا تغییرات اعمال بشه!

وقت اضافی داری؟ این متو بخون! لوکال هاست زمپ را نمیدانم یا شاید هم نمیتواند که دوباره از نو وصل شود تا تغییرات برگه اسکریپت در بروزر نمایان شود به همین دلیل بجای ان از این پکیج استفاده کردیم. در آینده میبینید که مدام با ctrl+c کار میکنیم تا از نو یک http اجرا کنیم تا تغییرات را مشاهده کنیم. شاید برای پروژه های دیگرم چنین کردم. یادم میاد حتی تغییرات فونت یا css هم با تاخیر اعمال میشد. دلیلش این بود که دیر به دیر رفرش میشد. اما در httpster رفرش کردن دست خودمان است. چه خوب شد باهاش آشنا شدم! احتمالا تبدیل میشه به یکی از دوستان صمیمی من!

هنوز وقت اضافی داری؟ راهنمای گام به گام راه اندازی React در سایت اصلی اش!
https://reactjs.org/docs/add-react-to-a-website.html

۲.۹ دستکاری ۱: افزودن استایل

شماره ۱: ببین اینطوریم درسته!

```
const{createElement}=React
const{render}=ReactDOM
```

شماره ۲: حالا یه ابجکت تعریف کنیم توش استایلو بریزیم که بعدا بدیمش به title

```
const style1 = {
  color: "yellow",
  backgroundColor: "green"}
```

شماره ۳: حالا المنت جدید بسازیم

```
const title = createElement("p",{...},"Hello")
```

شماره ۴: قسمت سه نقطه ابجکت اتریبیوت هاست:

```
{
  className: "class1",
  style: style1
}
```

در ابجکتی که اتریبیوت ها تعریف شدند، یک اتریبیوت پرحجم به نام style هست که از class و بقیه اتریبیوت ها بزرگتره بخاطر همین نمیشد مستقیما همه اطلاعاتشو یهو وارد کنم به داخل ابجکت اتریبیوت ها! اومدم در چند سطر پیش یک ابجکت دیگه به نام خودش ساختم و بهش گفتم برو خونه خودتون دراز بکش :)

بهش میگم تابع چون پرانتز جلوش داره!

```
render(... ,...)
```

تابع ()render دو مقدار اولیه میگیره:

۱. المنت ساخته شده

۲. آدرسش در برگه HTML

به این ترتیب شکمش پر میشه :)

```
render(title ,document.querySelector('#id1'))
```

یه نگاه کلی به محتویات index.js حالا محتویات index.js باید به این شکل باشد. همانطور که ملاحظه میکنید ۴ تا const داریم با یک render !

```
const{createElement}=React
const{render}=ReactDOM
const style۱ = {color: "yellow",backgroundColor: "green"}
const title = createElement("p",{className:"class1",style:style۱
    },"Hello")
render(title ,document.querySelector('#react-container'))
```

کدهای بالا در بروزر بالا میان اما میتونیم کدها را در رندر بریزیم. کاری که در دستکاری ۲ انجام دادیم.

۳.۹ دستکاری ۲: همه رو بریز تو ()render

```
const{ render }=ReactDOM
render(<p className='class1' style={{color: "yellow",
    backgroundColor: "green"}}> Hello </p>, document.
    querySelector('#react-container'))
```

اینجا باید ارور بده! میگه syntax اشتباهه! علامت > نباید باشه! بروزر نمیتونه ES۶ بخونه! باید از babel یا webpack استفاده کنیم. باید برای بروزر قابل خواندن بشه! ES۶ تبدیل بشه به ES۵ ! به این کار میگن کامپایل کردن Compiling یا Transpiling ! انجام این تبدیل از ES۶ به ES۵ با دو پکیج در Nodejs امکان پذیر است. البته خود Node هم بلده کامپایل ES۶ به ES۵ انجام بده!

۱.۳.۹ استفاده از سایت babel

کافیست کدهایتان را به لینک زیر ببرید و پیست کنید!

`https://babeljs.io/repl`

۲.۳.۹ استفاده از لینک CDN

۱. گوگل سرچ cdnjs babel-core و رفتن به اولین سایت

۲. انتخاب ورژن ۵.۸.۳۸ و کپی به یک تگ جدید script در برگه html

۳. تغییر اتریبیوت type در تگ script ای که index.js دارد به text/babel

۴.۹ اولین babel من :)

۱.۴.۹ پیش نیاز ها (۴ گام)

۱.۱.۴.۹ برگه html

۱. حذف لینک CDN بابل

۲. حذف تگ اسکریپت index.js

۳. ایجاد تگ جایگزینش به نام bundle.js با تایپ معمولی خودش

```
<script type="text/JavaScript" src="bundle.js"></script>
```

۲.۱.۴.۹ نصب babel-cli

۱. ادرس دهی محلی با cd
این ادرس برای تعیین محل استقرار فایل جیسونه! که در مرحله بعد قراره به جمعمون اضافه بشه!

۲. ساخت فایل جیسون با npm init
جیسون میسازیم همچین و همچون تا پکیج ها نصب بشن، اینجا برای نصب پکیج بابل داریم جیسون میسازیم!

۳. نصب babel-cli با دو خط

```
npm install --save-dev babel-cli
npm install -g babel-cli
```

۳.۱.۴.۹ پوشه بندی

در پوشه Rjs-html دو تا پوشه داریم به نام dist و src که در پوشه dist برگه html و در پوشه src برگه اسکریپت index.js قرار میگیره! در کنار دو پوشه یک فایل جیسون هم هست به نام package.json

۴.۱.۴.۹ نصب babel-preset-x (۲ گام)

۱. ایجاد فایل babelrc. در کنار package.json

در فایل babelrc. توجه کنید که قبل از پسوند هیچ نامی ندارد. فقط پسوند دارد. احتمالا از طریق windows-explorer قابل ایجاد نیست بنابراین در نرم افزار Atom ان را ایجاد میکنیم سپس ویندوز میپذیرد که چنین فایلی به دنیا بیاید!!

```
{'presets': ['latest','react','stage-0']}
```

۲. ترمینال: نصب

```
npm install --save-dev babel-preset-latest babel-preset-react babel-preset-stage-0
```

۲.۴.۹ اینم از اولین babel من!!

```
babel src/index.js --out-file dist/bundle.js
```

۳.۴.۹ تست بروزر (۲ گام)

۱. ترمینال - httpster

```
httpster -d dist -p 3000
```

۲. بروزر - localhost

به بروزر به ادرس localhost:۳۰۰۰ میرویم. اگه بالا اومد یعنی babel کارش درسته و کدهای ES۶ در فایل bundle.js به درستی به ES۵ کامپایل شده است. اینو گفتم بخاطر اینکه گاهی همون ES۶ را از index.js کپی میکنه بدون هیچ تغییری میریزه به bundle.js

۴.۴.۹ میانبر npm start - (اختیاری) (۲ گام)

به فایل جیسون میریم و به جای اسکریپتی که نوشته توش test مینویسیم start و تغییرات زیر را هم جلوی استارت وارد میکنیم:

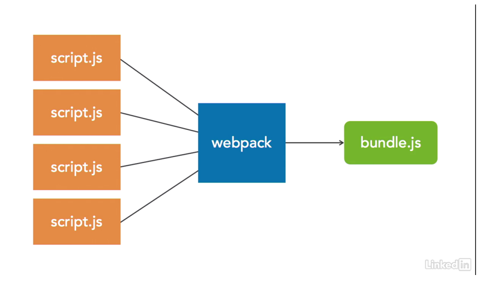
```
'start': 'httpster -d dist -p 3000'
```

حالا به ترمینال میریم و ...

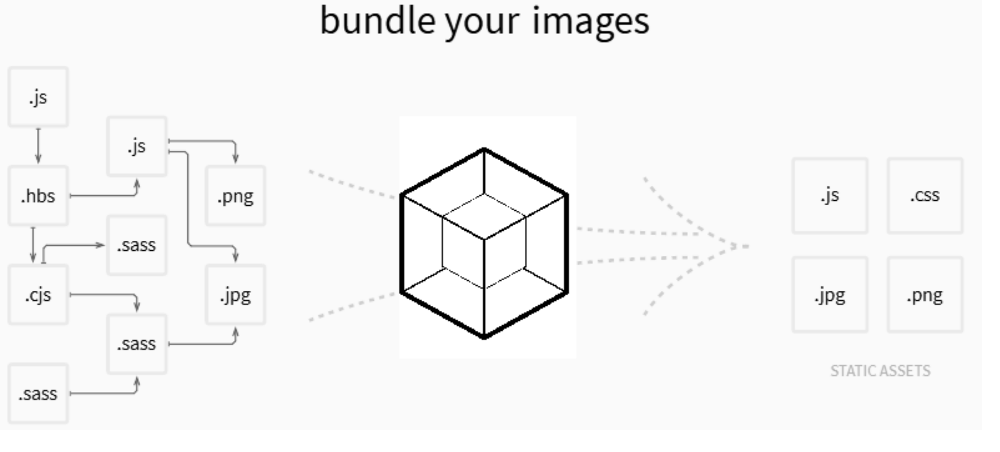
```
npm start
```

دیگه هرموقع خواستیم httpster راه بندازیم مینویسیم npm start خوشت اومد؟ 😲 حواسمون باشه این کد میانبر در مسیر Rjs-html کار میکند.

اگر حال داری این متنو بخون: یکی از مشکلاتی که توسعه دهندگان جاوااسکریپت با آن مواجه می شوند، تعدد بالای فایل های جاوااسکریپت، استایل ها و تصاویر، فونت ها و کلا asset هایی است که باید به صفحات وب شان لینک کنند. برخی مواقع تعداد این فایل ها به صدها فایل افزایش می یابد یک روش خوب برای حل مشکل، انجام باندلینگ است، ادغام و به هم وصل کردن این فایل ها باعث می شود که مرورگر فقط یک درخواست برای دریافت کلیه فایل ها ارسال و در پاسخ، یک فایل، حاوی کلیه اسکریپت ها یا استایل های مورد نظر دریافت کند. به این کار باندلینگ می گوییم. اما در فرایند باندلینگ یک نکته مهم دیگر هم وجود دارد، فایل ها و ماژول های جاوااسکریپت ممکن است که به همدیگر وابستگی داشته باشند و همچنین طبعا تاخر و تقدم قرار گیری هم مهم است. پس ما نیاز به یک ابزار باندلینگ هوشمند داریم که بتواند این وابستگی ها را به نحو مناسبی کشف و مدیریت کند. در این راستا ابزار های متعددی مانند گرانت، gulp، uglify و ... وجود دارند که برخی از این مشکلات را حل می کنند. ما نیاز به یک ابزار کامل داریم که همه این ویژگی ها را داشته باشد. وب پک (webpack) یک ماژول باندلر قدرتمند برای برنامه های کاربردی مدرن جاوا اسکریپت است. وقتی وب پک برنامه شما را پردازش می کند، به صورت بازگشتی، یک گراف وابستگی از هر ماژولی که برنامه شما نیاز دارد، می سازد. سپس تمام آن ماژول ها را در درون فایل های باندل های کوچکی (اکثرا یک ماژول) قرار می دهد که توسط مرورگر لود می شود. منبع ویکیپدیا.



وبپک مثل بابل است. شایدم داداششه! از کجا معلوم :)
شکل روبرو میگوید پکیج webpack میتواند همزمان چند فایل اسکریپت را کامپایل کند به یک فایل استاتیک به نام باندل!
`https://webpack.js.org/`



۵.۹ اولین webpack من :)

برای اولین تبدیل با وبپک نیاز به یک فایل با چند خط کد دارم. وبپک مثل بابل نیست که با یک خط کد در در ترمینال بتونه تبدیل کنه!

۱.۵.۹ پیش نیاز (۲ گام)

۱.۱.۵.۹ اماده سازی برگه html

اتریبیوت src اپدیت بشه به assets/bundle.js همین!

۲.۱.۵.۹ ترمینال - نصب

بازم مثل همیشه ممد برنده میشه :) حالا میرم مرحله اخر نصب این پکیج را در ترمینال بگذرونم!

با وارد کردن این کدهای آشنا:

```
cd Desktop/Rjs-html
npm install -g webpack
npm install webpack babel-
    loader webpack-dev-server
     --save-dev
```

۲.۵.۹ اینم از اولین وبپک من!!

باید یک فایل ایجاد کنم به نام webpack.config.js در کنار جیسون و درونش چند خط کد بیشتر از کدهایی که برای بابل در ترمینال وارد میکردیم بنویسیم تا تعیین کنیم از چه برگه ای بگیره و به چه برگه ای بریزه! آدرسشم بدیمو و از این جور حرفا!

```
var webpack = require("webpack");
module.exports={mode: 'production',
  entry: "/Users/shervin/Desktop/Rjs-html/src/index.js",
  output: {path:"/Users/shervin/Desktop/Rjs-html/dist/assets",
      filename:"bundle.js",publicPath:"assests"},
  devServer: {inline: true,contentBase: "dist",port: 3000},
  module:{rules:[{test: /\.js$/,exclude: /(node_modules)/,
 use: {loader: "babel-loader",
 query:{presets:["latest","react","stage-0"]}} }]}}
```

۱.۲.۵.۹ چجوری حفظش کنم؟

یک متغیر و یک ابجکت دارد. به همین سادگی! و آن ابجکت یک اتریبیوت به نام exports است که با یک نقطه از المنت module احضارش کردیم تا مقدارشو با کلید و value های جدید تغییر بدیم!
https://webpack.js.org/configuration/

```
var webpack = require("
    webpack");
module.exports={
  entry: ... ,
  output: ... ,
  devServer: ... ,
  module: ... ,
}
```

۳.۵.۹ تست بروزر (۳ گام)

۱.۳.۵.۹ ترمینال - وبپک

تست نمیخواد! چون من تستش کردم! حالا اگرم یه وقت خواستی برو ترمینال بزن webpack البته باید توی مسیر Rjs-html باشی! GoodLuck آسون بود نه؟ منم با تو هم عقیده ام

۲.۳.۵.۹ ترمینال - هاست

توی مسیر Rjs-html هاست httpster را راه اندازی میکنیم. حالا اگر از قبل روشن بود دو دکمه ctrl+c را میزنیم تا هاست آپدیت شود.

۳.۳.۵.۹ تماشا در browser

در بروزر به آدرس localhost:۳۰۰۰ بروید!

۴.۵.۹ زنگ تفریح بده به خودت!

یه چیزیم بگم این وسط! هرموقع خواستی بری صفاسیتی برو! قدم بزنی، موزیک و هوا عوض کردن و این حرفا :) وقتی تفریحی باشه بهترهم یاد میگیری وقتی برگردی! حافظه بلند مدتت هم خوب اینارو ته نشین میکنه به مغزت!

۵.۵.۹ فکر میکنید من به ارور نخوردم؟

این کدهایی که اینجا در این راهنما (شاید اسمشو بشه گذاشت کتابچه) نوشتم، حاصل رفع ارور ها بوده. قبلش کدهای اشتباهی بودن. اونارو حذف کردم و این کتابچه بسیار مورد ویرایش قرار گرفته! مثلا وقتی کد اشتباهی زدم و رفتم ترمینال و وارد کردم:

```
webpack
```

دیدم بله! همون ارورهایی که در کتابچه قبلی من به نام "پیچ شمرون با ES۶" میدادو داره میده! میگه اصلا ما loaders نداریم! رفتم به سایت اصلیش:
https://webpack.js.org/configuration/module/#ruleloaders
در لینک بالا نوشته دیگه از loaders استفاده نمیشه! قدیمی و منسوخه! بجاش باید چیکار میکردم؟ هیچی!! رفتم سراغ template خودش! تسلیمش شدم! گفتم بابا از همون اولم باید میومدم پیش خودت گام به گام بهم بگی چیجوری نصبش کنم! اما باز من نمیتونستم کل کارایی که کردمو بیخیال شم! دقت کردم! گفتم اصلا چرا باید از loaders استفاده میکردیم؟ برای فراخوانی babel-loader خب ... حالا سرچ میکنم ببینم بابل-لودر را چجوری خودش فراخوانی میکنه! سرچ کردم webpack babel-loader و نتیجه اش این لینک شد:
https://webpack.js.org/loaders/
بجای loaders از use استفاده میکنن!
ارور های دیگر هم اینا بودن:

- بابل-کور پیدا نشد
- ادرس دهی path باید absolute باشه
- مقدار loader باید استرینگ باشه
- loaders منسوخ شده
- چرا mode نداره؟
- ادرس دهی entry غلطه

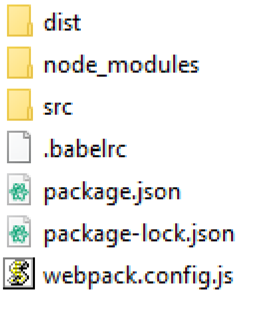
همه ارور هارو از سایت خودش جوابشو پیدا کردم بجز ادرس دهی که از سایت زیر پیدا کردم:
https://docs.microsoft.com/en-us/dotnet/standard/io/file-path-formats
همچنین پکیج بابل-کور را نصب کردم:

```
cd Desktop/Rjs-html
npm install babel-core --save-dev
```

فرقی نداره save-dev اولش باشه یا اخرش! :)

دو تا یاداوری کوچولو: هربار خواستی ادرس عوض کنی قبلش cd بزنیا :) هروقت خواستی نصب کنی قبلش npm بزنیا :)

پوشه های ما تا اینجا این شکلی هستن!



۶.۹ دومین webpack : بارگیری JSON

۱.۶.۹ پیش نیاز (۵ گام)

۱.۱.۶.۹ ترمینال - نصب

```
cd Desktop/Rjs-html
npm install --save react react-dom
npm install --save-dev json-loader
```

۲.۱.۶.۹ برگه html

دیگه به دو اسکریپت CDN - react نیازی نداریم چون پکیجشونو همین الان نصب کردیم.

۳.۱.۶.۹ برگه js

تمام برگه ما تغییر میکنه به این:

```
import React from "react"
import {render} from 'react-dom'
import {goodbye,hello} from './lib'

render(<div>{hello}{goodbye}</div>,
document.querySelector('#react-container'))
```

۴.۱.۶.۹ ایجاد فایل titles.json

یبار ارور داد که چرا بجای دبل کوتیشن، کوتیشن یدونه ای گذاشتی! یادم باشه توی فایل جیسون اجزای داخلی شبه شی را با دبل کوتیشن نشون بدم!

```
{"hello" : "Salamaleik", "goodbye" : "Khodahafez Aziz"}
```

۵.۱.۶.۹ ایجاد فایل lib.js

در پوشه src میخواهیم دو فایل ایجاد کنیم. یکی lib.js و دیگری title.json ! حالا داخل lib چی بریزیم؟ اینارو! بفرمایید!

```
import React from 'react'
import text from './titles.json'
export const hello = (<h1 id="title"
className="header"
style={{backgroundColor:'purple',color:'yellow'}}>
{text.hello}</h1>)
export const goodbye = (<h1 id="title"
className="header"
style={{backgroundColor:'yellow',color:'purple'}}>
{text.goodbye}</h1>)
```

چجوری حفظش کنیم؟ میشه اینطوری حفظش کرد!

```
import React from 'react'
import text from './titles.json'
export const hello = (...)
export const goodbye = (...)
```

داخل سه نقطه یک تگ h1 قرار میگیره! این تگ id داره و استایل و کلاس نیم و یک شی که در شکمش قرار گرفته:

```
<h1 id="title"
className="header"
style={{backgroundColor:'yellow',color:'purple'}}>
{text.goodbye}</h1>
```

به همین راحتی قابل حفظ کردنه مگه نه؟ خیلی هم دوست داشتنیه! ❤

۲.۶.۹ اینم از بارگیری جیسون!!

نیازی به وارد کردن کد به webpack.config.js نداریم! بصورت پیش فرض json-loader فعال است. هورااا تنها نکته اینه که bundle.js که در پوشه asset هست ساخته بشه که میریم ترمینال میزنیم webpack

۳.۶.۹ تست بروزر (۲ گام)

۱. هاست npm start

۲. برای مشاهده در browser به ادرس localhost:۳۰۰۰ میرویم تا ببینیم چی شده دومین کد react خوشگلمون 😛

۴.۶.۹ فکر میکنید من به ارور نخوردم؟

۱.۴.۶.۹ سرچ ۱:

توی گوگل سرچ کردم json-loader این دو لینک از دو سایت مرجع، حواسمو پرت کرد! رفتم ببینم توش چخبره! دیدم هردوتاش یکیه! احساس میکنم باید کپی پیست کنم.
https://www.npmjs.com/package/json-loader
https://webpack.js.org/loaders/json-loader/

۲.۴.۶.۹ سرچ ۲:

برای بار دوم سرچ کردم جواب داد! انگار باید دوبار سرچ کرد بیار کافی نیست! گاهی هم باید انقدر تست کنم تا یاد بگیرم. هورراا پیدا شد! توی github دربارش پست گذاشتن. نیازی نیست کد وارد کنیم! بصورت پیش فرض خودش json-loader اجرا میکنه! اگه کد نوشتیم دوبار اجرا میشه و ارور میده! کدهای webpack.config.js بایستی حذف بشن! البته اینو توی سایت خودشم نوشته بود منم فهمیدم چی میگه اما من دقت نکردم که باید حذفش کنم. من با حذف کردن مخالفم آخه همیشه :-)
https://stackoverflow.com/questions/49358313/json-loader-in 49373676#49373676

۷.۹ سومین webpack : اضافه کردن css به webpack-build

۱.۷.۹ پیش نیاز (۵ گام)

۱.۱.۷.۹ اماده کردن lib.js

ابتدا به lib.js میریم و دو تا style را پاک میکنیم. بجاش در className ها hello و در دیگری goodbye مینویسیم. بعدش این دو خط را وارد میکنیم. در اینده قراره این دو فایل ساخته بشن!

```
import './stylesheets/hello.css'
import './stylesheets/goodbye.scss'
```

۲.۱.۷.۹ پوشه بندی

در پوشه src یک پوشه به نام stylesheets ایجاد میکنیم و در آن دو فایل به نام های hello.css ، goodbye.scss ایجاد میکنیم.

۳.۱.۷.۹ ایجاد فایل hello.css

```
.hello{background-color: #aaa; color: yellow}
```

۴.۱.۷.۹ ایجاد فایل goodbye.scss

```
$bg-color: yellow;
$text-color: #aaa;

.goodbye{
  background-color: $bg-color;
  color: $text-color}
```

۵.۱.۷.۹ ترمینال نصب

توی مسیر لوکال باشیم

```
npm install css-loader style-loader sass-loader node-sass --save-dev
```

۲.۷.۹ اینم از لود کردن استایل در وبپک!!

در module.exports.module.rules بعنوان دومین عضو آرایه، این ابجکت را قرار میدهیم:

```
{test: /\.css$/i,
use: ['style-loader','css-loader'],},
{ test: /\.scss$/i,
 use: ['style-loader','css-loader','sass-loader']}
```

فکر نکنید اشتباه میگما اون آخرین لودر دقیقا sass هست نه scss !
درگوگل سرچ شود loaders-webpack انگاه به سایت اصلی رفته شود. در لیست loader ها autoprefixer-loader نداشت بخاطر همین حذفش کردم، دیگه ارور نداد! هوررا ... :) اموزش لیندایی که میبینیم احتمالا مال وبپک ۳ یا وبپ پک۲ درحالیکه الان با وبپک ۴ کار میکنیم.
https://webpack.js.org/loaders/css-loader/
https://webpack.js.org/loaders/style-loader/

۳.۷.۹ تست بروزر (۳ گام)

- ترمینال webpack : جهت ساخت فایل استاتیک bundle.js که در فایل html فراخوانی شده!
- ترمینال هاست با start
- بروزر به ادرس localhost:۳۰۰۰

هورراااا... :) این فصل تموم شد! بریم فصل ۲ با اولین کمپوننت ها :)

صبح بخیر کمپوننت ها!

آنچه در این فصل میخوانید: اینجا به ۴ حالت component میسازیم. موقع فراخوانی مثل المنت element های html هستند! چون ReactDOM زورش زیاده میتونه المنت بسازه و به DOM اضافه کنه! نام المنت همنام کمپوننت است که با حرف اول بزرگ نوشته میشود. المنت های html اتریبیوت داشتن! مثلا تگ img (یا همون المنت img) چه اتریبیوت هایی داشت؟ src ، width ، height و alt ! حالا اینجا المنت، اتریبیوتی داره که اسمش میشه propperty یا props ! مقدارشون داخل براکت-منحنی قرار میگیره نه مثل اتریبیوت های المنت html که داخل دبل کوتیشن قرار میگرفت. میخوایم بگیم props میتونه با this بیاد یا تنها بیاد یا اصلا بره تو کلاس! اونوقت بجاش state هم میتونه بیاد! خلاصه اینکه اتفاقای جالبی قراره بیفته! کلی خوش بگذرونیم و لذت ببریم! در سرزمین Reactjs کمی جلوتر قدم میزنیم به getDefaultProps و داداشش defaultProps میرسیم. بعدش با بقالی سرکوچه شون propTypes آشنا میشیم. 😀

گیمیفیکیشن با React : مثل font-awesome میمونه! اسم پکیج react-icons است. نصب کن بعد با خطوط زیر ایمپورت کن! حالا سه المنت داریم که هرجا در متن بذاریم همونجا ایکون ظاهر میشه!

```
import { MdAccessibility } from "react-icons/md";
import { MdAndroid } from "react-icons/md";
import { MdExtension } from "react-icons/md";

<MdExtension/>  <MdAndroid/>  <MdAccessibility/>
```

از کجا فهمیدم این کدارو باید بذارم؟ از سایت پکیج react-icons

https://react-icons.netlify.com/#/

۱.۱۰ ساخت component (۲ روش ۴ حالت)

- ساخت component با تابع js (۲ حالت)
  - با props
  - با براکت منحنی
- ساخت component با کلاس (۲ حالت)
  - کلاس ES۶
  - کلاس React

هر ۴ تا حالت در بروزر به یک شکل نمایش داده میشوند فقط در قاعده نوشتاری با هم فرق دارند. در ادامه با شکل توضیحش دادیم!

۱.۱.۱۰ حالت اول

```
//part 0
import React from 'react'
import {render} from 'react-dom'
//part 1
var Fn1 = (props) =>
<h1>This is {props.key01} from Mohammad Amin! </h1>
//part 2
render( <Fn1 key01={"Hello World!"} />,
document.querySelector('#react-container') )
```

https://reactjs.org/docs/components-and-props.html
reactjs.org > Docs > Main Concept > Components And Props

نکته: وقتی یک تگ html میاد به React دو تفاوت داره!

۱. بجای class از className استفاده شده!

۲. پروپ ها بجای دبل کوتیشن، براکت-منحنی دارند!

۲.۱.۱۰ حالت دوم

```
const Fn1 = ({key01}) => <h1>This is {key01}</h1>
...
render( <Fn1 key01={value01}/> , ... )
```

۱.۲.۱.۱۰ متد ها

به داخل کلاس اضافه میکنیم بطوریکه قبل از render قرار بگیره!

```
fn1(x5){return((x5*100)+'%')},
fn2(x1,x2){return this.fn1(x1/x2)},
```

داخل اخرین div یک span هست که داخلش این کد را میریزیم:

```
{this.fn2(this.props.x1,this.props.x2)}
```

تابع بدون state حالا لوکال متد ها یعنی fn1 و دوستش را به قبل از کد بالا میبریم. تنها فرق این دو تابع کانست دار با قبلیا نداشتن this در شکمشون هست.

```
const fn1 = (x5) => {return((x5*100)+'%')}
const fn2 = (x1,x2) => {return fn1(x1/x2)}
```

۳.۱.۱۰ حالت سوم

```
class Greeting extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

۴.۱.۱۰ تبدیل کلاس ریاکت به کلاس ES۶ :

اختیاری: علامت کاما بعد از متد fn۱ ، fn۲ نداره!
اختیاری: میشه فقط Component ایمپورت کرد! 😀

```
import {Component} from 'react'
... ...
... extends Component{...
```

اختیاری: مثل کلاس react پرانتز نداره!

```
export class X extends React.Component{...}
```

۵.۱.۱۰ حالت چهارم

```
var createReactClass = require('create-react-class');
var Greeting = createReactClass({
  render: function() {
    return <h1>Hello, {this.props.name}</h1>;
  }
});
```

نکته: اگه در کنسول لوگ ارور داد react نمیشناسیم یک خط از بین این دو تا راه حله!

```
window.React = React
import React from 'react'
```

نکته: حرف اول کمپوننت بزرگ باشد! نکته: درلحظه فراخوانی کمپوننت، المنت با یک تگ یا دو تگ فرقی ندارد.

```
<X /> or <X></X>
```

۶.۱.۱۰ پروپ props

به قطعه کد زیر نگاه کنید! x۱ ، x۲ ، x۳ ، x۴ داخل المنت X پروپرتی هستند. انها مقدار اولیه یا ورودی تابع سازنده کمپوننت هستند.

```
<X x1={10} x2={100} x3={30} x4={23}>
```

در خط بالا فقط عدد آمده. اگر string یا دیگر مقادیر را بخواهیم بگذاریم از لحاظ سینتکس قاعده نوشتاری اینگونه نوشته میشود:

```
x1={true}
x1={"Mohammad Amin"}
x1={10}
x1={x100}
```

نکته: وقتی پروپرتی استفاده کنیم، اعداد بصورت داینامیک می آیند! داینامیک در مقابل استاتیک! استاتیک برای تابع بدون ورودی است که مثلا یک متن ساده یا اعداد ثابتی را برمیگرداند.

پیشخوانی بخش بعدی: هرچقدر جلوتر میرویم قواعد نوشتاری (بدون تغییر نمایش در بروزر) بهبود پیدا میکند! اینجا قرار است کار به جایی برسد که المنت ساخته شده تنها نامش بیاید تا فراخوانی شود. پروپ هایش داخل کلاس تعریف شوند. state امده است این کار را بکند.
در این مثال، ثانیه و دقیقه و ساعت در بروزر نمایش داده میشود. بروزر تغییرات لحظه ای را نشان میدهد. اولش با تابع تعریف میکنیم! بعد کلاس میشه! بعد بهش state اضافه میشه! همشونم درستن! چه تابع! چه کلاس بدون state ! چه کلاس با state ! فقط سینتک بهبود پیدا میکند! کارمون راحت میشه! دیگر لازم نیست پروپرتی بدیم هربار که المنت تعریف میکنیم. از مبدا با state مشخص میشه!
تابع کمپوننت ساز به کلاس ES۶ تبدیل میشود! همان کلاسی که کنستراکتور دارد! و همانطور که میدانید کنستراکتور state دارد! دو رفیق پیدا میکنیم! this.setState و this.state که به جماعت سرزمین React اضاف میشن!

۲.۱۰ بیگ پروژه ۱: State

```
var fn = () => {
var Clock = ({date}) =>
<div><h1>Hello World!</h1>
<h2>It is {date.toLocaleTimeString()}!</h2></div>

render(<Clock date={new Date()}/>
,document.querySelector("#react-container") ) }

setInterval(fn , 1000)
```

😍 پس از نوشتن کمپوننت و رندر، هر دو را داخل تابع fn ریختم و fn هم در setInterval قرار گرفت.

۱.۲.۱۰ تابع میشه کلاس!

یادآوری: این سه خط جیگر هم هستن! همش همونا‌س فقط کمپوننت به تنهایی ایمپوت شده!

```
import React from 'react'
import {Component} from 'react'
import {render} from 'react-dom'
```

یه کلاس وراثتی ES۶ میخوایم بسازیم: مرحله اول: داخل کلاس متد render با محتویاتی که تابع داشت میریزیم. مرحله اخر: this به اول props اضافه میکنیم.

```
class Clock extends Component {
render = () =>
<div><h1>Hello, world!</h1><h2>
It is {this.props.date.toLocaleTimeString()}.</h2></div> }

var fn = () => {
render( <Clock date={new Date()} />,
document.querySelector('#react-container') ) }

setInterval(fn, 1000);
```

۲.۲.۱۰ کلاس state دار میشه!

وقتی constructor به کلاس وراثتی اضافه بشه، خودبخود state بهش اضافه میشه!

۱. اضافه کردن constructor به کلاس

```
constructor(props){super(props);
this.state = {date: new Date()} }
```

۲. تبدیل this.props به this.state

```
{this.state.date.toLocaleTimeString()}
```

۳. حالا میشه بدون پروپ از کمپوننت استفاده کرد

```
render( <Clock />, ...
```

۳.۲.۱۰ چند نکته

۱.۳.۲.۱۰ لوکال state در کلاس ES۶

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = {count: props.initialCount};
  }
  // ...
}
```

۲.۳.۲.۱۰ لوکال state در کلاس React

```
var Counter = createReactClass({
  getInitialState: function() {
    return {count: this.props.initialCount};
  },
  // ...
});
```

۳.۳.۲.۱۰ نکته:

در render و constructor از this.state استفاده میشه و جاهای دیگه this.setState !

۴.۳.۲.۱۰ نکته:

داداشمون this.setState بدون تابع مهمونی نمیاد!

```
// Wrong
this.setState({
  counter: this.state.counter + this.props.increment,
});
```

براش تابع میذاریم!

```
// Correct
this.setState((state, props) => ({
  counter: state.counter + props.increment
}));
```

۵.۳.۲.۱۰ نکته:

میشه عکس نکته قبلیو با تابع ES۵ هم نوشت!

```
// Correct
this.setState(function(state, props) {
  return {
    counter: state.counter + props.increment
  };
});
```

۶.۳.۲.۱۰ نکته:

اگر چند متغیر داشته باشیم state اینگونه میشود

```
constructor(props) {
  super(props);
  this.state = {
    posts: [],
    comments: []
  };
}
```

و در ادامه:

```
componentDidMount() {
  fetchPosts().then(response => {
    this.setState({
      posts: response.posts
    });
  });

  fetchComments().then(response => {
    this.setState({
      comments: response.comments
    });
  });
}
```

۴.۲.۱۰ ادامه: اضافه کردن lifecycle

دو تا متد بهش اضافه شده، تابع fn هم به داخل اورده شده، همین!

```
//part 0
import React from 'react'
import {Component} from 'react'
import {render} from 'react-dom'
//part 1
class Clock extends Component {
  constructor(props){super(props); this.state = {date: new Date()}}
  componentDidMount(){this.timerID = setInterval(()=>this.fn(),1000)}
  componentWillUnmount(){clearInterval(this.timerID)}
  fn(){this.setState({ date: new Date() }) }
  render = () => <div><h1>Hello, world!</h1>
    <h2>It is {this.state.date.toLocaleTimeString()}.</h2></div>}
//part 2
render( <Clock />,
document.querySelector('#react-container') )
```

اینجوری هم میشه!

```
function App() {
  return (
    <div>
      <Clock />
      <Clock />
      <Clock />
    </div>
  );
}

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

۳.۱۰ بیگ پروژه ۲: Extract

در سایت ری اکت کمپوننت بزرگتر را تکه تکه کرده و به ان extract گفته است.

چرا extract میکنیم؟ میخوایم بارها یه المنتو فراخوانی کنیم. بخاطر همین فقط در یک کمپوننت تعریفشون نمیکنیم بلکه خارج از ان تعریف کرده و داخل ان فراخوانی میکنیم مثل همه جاهای دیگه ای که بعدا میخوایم فراخوانی کنیم.

۱.۰.۳.۱۰ یه کمپوننت بزرگ بسازیم تا تجزیه اش کنیم!

```
1  import React from 'react'
2  import { render } from 'react-dom'
3  
4  const Comment = ({props}) =>
5  <div className="Comment">
6  <div className="UserInfo">
7  <img
8  className="Avatar"
9  src={props.author.avatarUrl}
10 alt={props.author.name} />
11 
12 <div className="UserInfo-name"> {props.author.name} </div>
13 </div>
14 
15 <div className="Comment-text"> {props.text} </div>
16 <div className="Comment-date"> {props.date} </div>
17 </div>
18 
19 var key01 = {
20 author: {
21 avatarUrl:"#",
22 name:"Mohammd Amin"
23 },
24 text: "dsfsdfsdfsdfsdf",
25 date: "2/2/2019" }
26 
27 render(
28 <Comment props={key01} />
29 ,document.getElementById('react-container') )
```

اگر توجه کنید در خط ۵ ، ۶ ، ۱۲ و ... مقدار جلوی className مانند اتریبیوت های تگ html داخل دبل کوتیشن نوشته شده اما در html بجای className داشتیم class 😛 اینم شباهت و تفاوتش بود با html .

۲.۰.۳.۱۰ اولین extract : اولین بچه-کمپوننت

برای کباب کردن (اکسترکت کردن) یکی از کمپوننت های بالا مثل avatar کد های part۱ به دو بخش زیر تقسیم میشود.

```
const Comment = ({props}) =>
<div><div>
<Avatar user={props.author}/>
<div> {props.author.name} </div>
</div>
<div> {props.text} </div>
<div> {props.date} </div>
</div>

const Avatar = ({user}) =>
<img
className="Avatar"
width="20px"
src={user.avatarUrl}
alt={user.name} />
```

۳.۰.۳.۱۰ دومین extract و به دنیا امدن دومین بچه-کمپوننت

با extarct کردن یک کمپوننت کوچکتر از کمپوننت بزرگتر میتوان ان کمپوننت کوچک را برای ساختن کمپوننت های بزرگ دیگر ارجاع داد. اجازه بدید یه کمپوننت دیگه بسازیم. تمام اطلاعات کاربر در این کمپوننت باشه! هرجا خواستیم اطلاعات بیاریم فقط یک کمپوننت بنویسیم.

```
const Comment = ({props}) =>
<div>
<UserInfo user۱={props.author} />
<div> {props.text} </div>
<div> {props.date} </div>
</div>

const Avatar = ({user}) =>
<img
className="Avatar"
width="20px"
src={user.avatarUrl}
alt={user.name} />

const UserInfo = ({user۱}) =>
<div>
<Avatar user={user۱} />
<div> {user۱.name} </div>
</div>
```

شاید این متنو ویرایش کنم و در ابتدا کمپوننت ها را تعریف بعد یکی بزرگشونو بسازم!

۱.۳.۱۰ تعیین مقدار پیشفرض defaultProps

چهار روش میشناسم

- یادته در ورودی تابع جاوااسکریپت مساوی داشتیم
- درون کلاس ری اکت از getDefaultProps استفاده میکنیم
- بعد از کلاس ری اکت از پروپرتی defaultProps که برابر ابجکت میشه استفاده میکنیم.
- جدیدا فهمیدم از static قبل defaultProps داخل کلاس ری اکت هم میشه استفاده کرد

https://reactjs.org/docs/typechecking-with-proptypes.html#default-prop-values

۱.۱.۳.۱۰ اعتبارسنجی در کلاس ES۶

```
class Greeting extends React.Component {
  // ...
}

Greeting.defaultProps = {
  name: 'Mary'
};
```

۲.۱.۳.۱۰ اعتبارسنجی در کلاس React

```
var Greeting = createReactClass({
  getDefaultProps: function() {
    return {
      name: 'Mary'
    };
  },
```

۲.۳.۱۰ اعتبار سنجی typechecking

نکته: پکیج داره! prop-types

اعتبارسنجی به ۳ روش:

- داخل تابع با propTypes ابجکت
- بعد از تابع با propTypes ابجکت
- دستی با propTypes

۱.۲.۳.۱۰ اعتبار سنجی دستی

اینجا از if استفاده میکنیم و متنی اروری که در کنسول لوگ نمایش داده میشه رو خودمون مینویسیم.
به خودکفایی رسیدیم!!

۲.۲.۳.۱۰ اطلاعات بیشتر

```
https://reactjs.org/docs/typechecking-with-proptypes.html
https://reactjs.org/docs/react-without-es6.html#declaring-default-props
```

۴.۱۰ بیگ پروژه ۳: table بسازیم!

یک جدول ساده مثل این را کدنویسی میکنیم!

Key1	Key2	Key3	Key4
value1-1	value2-1	value3-1	value4-1
value1-2	value2-2	value3-2	value4-2
value1-3	value2-3	value3-3	value4-3
value1-4	value2-4	value3-4	value4-4
value1-5	value2-5	value3-5	value4-5

هورراااا... 😮 بهتر تر از این نمیشه! تازه یه فیلمم ازش تهیه کردم! از نوشتن خودم روی کاغذ فیلم گرفتم و توی اسمش نوزده شهریور وجود داره! ولی توصیه میکنم نبینید چون حجمش زیاده!

```
//part 0
import React from 'react'
import { render } from 'react-dom'
window.React = React
//part 1 : sakhteman e table
const Fn2 = ({key01}) =>
<table>
  <thead><tr> <th>Key1</th><th>Key2</th><th>Key3</th><th>Key4</th> </tr></thead>
  <tbody>{key01.map( Fn1 )}</tbody>
</table>
const Fn1 = ({key1,key2,key3,key4}) => <tr> <td>{key1}</td><td>{key2}</td><td>{key3}</
//part 2 : meghdardehi e table
var value01 = [
{key1:'value1-1',key2:'value2-1',key3:'value3-1',key4:'value4-1'},
{key1:'value1-2',key2:'value2-2',key3:'value3-2',key4:'value4-2'},
{key1:'value1-3',key2:'value2-3',key3:'value3-3',key4:'value4-3'},
{key1:'value1-4',key2:'value2-4',key3:'value3-4',key4:'value4-4'},
{key1:'value1-5',key2:'value2-5',key3:'value3-5',key4:'value4-5'},
]
render(  <Fn2 key01={value01}/> , document.querySelector('#react-container')  )
```

فصل ۱۱

روت Route

در این فصل اشنا میشویم با Route ها!! توی ادرس بروزر نقش دارن! در این فصل با مباحث زیر اشنا میشیم!

react-router , react-router-dom
BrowserRouter , Router , Link
nesting Routes , match

۱.۰.۱۱ چند نکته پراکنده

نکته: برای اینکه ارور ۴۰۴ برای زیردامنه ی نامرتبط بیاد، از exact استفاده میکنیم. (آخه مادرش بصورت پیشفرض میومد)

نکته: برای اینکه اگر صفحه ای تعریف نشده بود بره روی ارور ۴۰۴ ، از Switch استفاده میکنیم.

نکته: روت زیردامنه بعد روت مادر بیاد. روت ارور باید اخرین روت باشه!

روش جمع اوری اطلاعات: برخی اطلاعات این کتابچه در لیندا نبودند و سرچ گوگل در بلاگ های سایتها کمکم کرد.

note: All Route props (match, location, history) are available to its Component (User).

```
function User(props) {
return <h1>Hello{props.match.params.userName}</h1>}

<Route path="users/:userName" component={User} />
```

۱.۱۱ اولین روت من :)

پوشه Rjs-html را حذف کردم. دیگر نیازی به webpack و ... نداریم. همگی را الان با یه سری چیزای دیگه نصب میکنیم. ابتدا وبپک را حذف نصب میکنیم و فایل های webpack.config.js را حذف میکنیم.

```
npm uni webpack
npm uni -g webpack
npm uni -g webpack-cli
npm uni webpack-cli
```

حالا نصب میکنیم که نسبتا زمان بره! بایستی صبور باشیم!

```
cd Desktop
npx create-react-app rjs-html
```

https://create-react-app.dev/

حالا یه پوشه rjs-html داریم و بجای اینکه هربار برای تست بروزر webpack ، npm-start بزنیم کافیست یکبار npm-start را در مسیر rjs-html بزنیم! هر دو کار را انجام میدهد! هوررااا ... داریم پیشرفت میکنیما! 😲

```
npm install react-router-dom --save
```

این فایل index.js

```
import React from 'react';
import {render} from 'react-dom';
import {BrowserRouter,Route} from 'react-router-dom'
import {E404} from './err404'
import {Home} from './home'

render(
<BrowserRouter>
<Route exact path="/" component={Home}/>
<Route path="*" component={E404}/>
</BrowserRouter>
, document.getElementById('root'));
```

یه فایل دیگه:

```
import React from 'react'

export const E404 = () =>
<h1>Error404</h1>
```

یه فایل دیگه:

```
import React from 'react'

export const Home = () =>
<h1>Home</h1>
```

همیجوری که ادیت میکنم در حالت npm-start خودش میفهمه اعمال میکنه در بروزر دمش گرم! نوکرتم بامرام!! عشق منی!! ❤️

کدهای بالاییو با پکیج react-router-dom نوشتم اما میشه از پکیج react-router استفاده کرد. Router بجای BrowserRouter میشه نوشت! بایستی یک prop به نام history بهش بدیم. بدین منظور متغیری همنامش تعریف کردم و از createBrowserHistory در پکیج history استفاده کردم. (این پکیج نیازی به نصب نداره!)

https://reacttraining.com/react-router/core/api/Router/history-object

```
//—— react
import React from 'react';
import {render} from 'react–dom';
//—— router
import {Switch, Route, Router} from 'react–router'
import { createBrowserHistory } from "history"
//—— components
import {E404} from './err404'
import {Home} from './home'
import {Contact} from './contact'
import {About} from './about'

const history = createBrowserHistory()
render(
<Router history={history}>
<Switch>
<Route exact path="/" component={Home}/>
<Route path="contact–us" component={Contact}/>
<Route path="/about–us" component={About}/>
<Route component={E404}/>
</Switch>
</Router>
, document.getElementById('root') )
```

بدیهی دو تا چیز: یکی اینکه دو کمپوننت مثل اون دوتا قبلی ساختم. یکی دیگه هم اینکه بروزر نیازی به رفرش هم نداره! چون از اون اول در مسیر، npm-start استفاده کردیم و تا الان هم قطعش نکردیم.

فقط یه چیزی اگر exact و Switch نباشن Home ، E۴۰۴ در تمام صفحاتی که با ادرس دیگری میایند ظاهر میشوند.

۲.۱۱ بچه محلای Route

۱.۲.۱۱ لینک

```
<Link to="/go" activeClassName="class1">Go</Link>
```

https://reacttraining.com/react-router/web/api/NavLink

۲.۲.۱۱ بچه و رندر

```
<Route exact path="/" render={ () => <div>item id 2 </div> }/>
<Route exact path="/" children={ () => <div>item id 2 </div> }/>
// we can use these instead
render={ props => <Item {...props}/> }
children={ props => <Item {...props}/> }
```

۳.۱۱ خرده پروژه: pathname

۱.۳.۱۱ مقدمه

اگر در index.js چند روت با چند path مختلف داشته باشیم بطوریکه همه روت ها یک component یکسان داشته باشند، همچنین به بروزر برویم به سه ادرس مختلف و سه کمپوننت مختلف ببینیم انگاه معلوم میشود که ان کمپوننت تکرار شده یک کلاس ES۶ بزرگ با چند بچه کمپوننت است. بچه کمپوننت ها با this.props.location.pathname فراخوانی میشوند.

۲.۳.۱۱ برگه index.js

```
<Route exact path="/" component={Child}/>
<Route path="/go" component={Child}/>
<Route path="/lab" component={Child}/>
```

۳.۳.۱۱ برگ app.js

```
const Home = () => <h1>Home</h1>
const Go = () => <h1>Come with Me!</h1>
const G1 = () => <h1>Come with Me! Come with Me!</h1>
export class Child extends Component {
  render() {return (
      <div>{
        (this.props.location.pathname === "/") ?
        <Home/> :
        (this.props.location.pathname === "/go") ?
        <Go /> : <G1 />
    }</div>
      ) } }
```

۴.۱۱ سلام match !

Hello match :) match is here to help us passing props to React Router's Link component. match is an object and is useful when you asked yourself 'how a path matched a url'.

match Properties

- url
- path
- params
- isExact

note: You can also find match properties in inspect element!

How match can be used?

- withRouter –> this.props.match
- Route component –> this.props.match
- Route child –> (match) => (...)
- Route render –> (match) => (...)
- matchPath –> as the return value

https://reacttraining.com/react-router/web/api/match

۵.۱۱ بیگ پروژه ۱: match

۱.۵.۱۱ مقدمه: سه قطعه کد اصلی اینان!

اولین قطعه خودش دو قسمت است. تعریف و مقداردهی.

```
<td>{key3 ? "good" : null}</td>
//
key1: "value11",
key2: "value12",
key3: false,
```

کلاس اکما ۶ میتواند کمپوننتی را در رندر خود جای دهد.
بدین ترتیب میتوان پروپ های روت یعنی match.params را به App و از App هم به کمپوننت درون ان نسبت داد

```
export class App extends React.Component{
render(){ return(<div><Fn1 keyo1={valueo1}
filter1={this.props.match.params.x}/></div>)}}
```

بنابراین الان کمپوننت درونی هم همان پروپ های روت را دارد. الان بهم مرتبط شدن اون path و این x که درون filter1 نوشته شده

```
<Route exact path="/d/:x" component={App} />
```

همچنین باید return بیاد. return هم داخل براکت منحنی خوانا میشه!

۲.۵.۱۱ عکس یادگاری

تمام کدها را یکجا ببینید! دو فایل هستند.

۱.۲.۵.۱۱ فایل اول به نام index.js

```
import React from 'react'
import { render } from 'react-dom'
import { Err,App} from './app'
import {Switch,BrowserRouter,Route} from 'react-router-dom'
window.React = React

render(
  <BrowserRouter>
  <Switch>
  <Route exact path="/" component={App} />
  <Route exact path="/:x" component={App} />
  <Route component={Err} />
  </Switch>
```

```
    </BrowserRouter>
    ,document.getElementById('root')
)
```

۲.۲.۵.۱۱ فایل دوم به نام app.js

```
import React from 'react'
import {Link} from 'react-router-dom'

export const Fn1 = ({key01, filter1}) =>{
  const X1 = (!filter1 || !filter1.match(/key3|key2/)) ?
  key01 : key01.filter( x => x[filter1] )
return (
<table>
  <thead>
    <tr>
      <th>Key1</th>
      <th>Key2</th>
      <th>Key3</th>
    </tr>
    <tr>
    <td colSpan={3}>
    <Link to="/">All</Link> ...
    <Link to="/key2">Key2</Link>...
    <Link to="/key3">Key3</Link>
    </td>
    </tr>
  </thead>
  <tbody>
    {X1.map(Fn2)}
  </tbody>
</table>
)}
const Fn2 = ({key1,key2,key3}) =>
<tr>
<td>{key1}</td>
<td>{key2 ? "lala" : null}</td>
<td>{key3 ? "good" : null}</td>
</tr>

export class App extends React.Component{
  render(){return(
<div>
<Fn1 key01={value01} filter1={this.props.match.params.x}/>
</div>)}
}

var value01 = [
  {
    key1: "value11",
    key2: false,
    key3: false,
  },
  {
    key1: "value21",
    key2: false,
    key3: false,
  },
  {
    key1: "value31",
    key2: true,
    key3: false,
  },
  {
    key1: "value41",
    key2: true,
    key3: false,
```

```
  },
  {
    key1: "value51",
    key2: false,
    key3: true,
  },
]

export const Err = () =>
<div>
<h1>Error 404!</h1>
Page Not Found!
<button><Link to="/">Back Home</Link></button>
</div>
```

Warning: Each child in a list should have a unique "key" prop.
https://fb.me/react-warning-keys
در لینک بالا که هنوز نرفتم و میخوام برم چون الان نت ندارم میخوام بفهمم که چطور باید از کلید های مختلف استفاده کرد. اخه اگه مختلف بشن دیگه کار نمیکنه!
این ارور در کنسول لوگ نمایش داده شد ولی بروزر همه چیو خوب اجرا میکرد.

موقع نستینگ مادرش در زیردامنه میاد چرا؟ عیبی نداره. نستینگ برا شبه Ajax خوبه مثلا -mor einfo ، فراخوانی چند شی تو در تو یا آپدیت کردن موارد داخل جدول

مستقیم زیردامنه بزنی در بروزر نمیاره چرا؟ اینکه خوبه چون بصورت موقت برای موارد خاص میخوایمش نه همیشه. مثلا تغییر موارد داخل جدول!

متچ اونجا وارد شد اینجا نشد؟ متچ در بیگ پروژه دو، ورودی تابع ها قرار گرفته و کاملا درست عمل کرده اما اینجا میزنم اثری نداره! چرا؟

نکته: اگر میخواستیم زیردامنه را با x: در یک روت در ایندکس بگذاریم انگاه برای زیر دامنه های نامرتبط مادر میامد بجای ارور.

۶.۱۱ سلام nesting !

دو قطعه کد جدیدش همینه! یکی match.url و دیگری match.path !

```
Link--> to={`${match.url}/shoes`}
Route--> path={`${match.path}/categoryName`}
```

مثال ساده

```
export class App extends Component{
  render(){return( <div> gogogog
    <BrowserRouter> <div>
      <Link to={`${this.props.match.url}/key3`}> Key3 </Link>
      <Route path={`${this.props.match.path}/:x`} render={
({match}) => <Fn1 key01={value01} filter1={match.params.x}/>
      }/> </div>
    </BrowserRouter>
    </div> )} }
```

نکته: متچ را میتوان در کمپوننت های تو در تو هم تعریف کرد. به معنی اینکه اگر در Fn1 یک روت بیاد که کمپوننتی داشته باشه انگاه هنگام تعریف ان کمپوننت در قسمت ورودی میتوان متچ وارد کرد. مثل مثالی که در بیگ پروژه نوشتم.

۷.۱۱ بیگ پروژه ۲: Nesting-Routes

۱.۷.۱۱ مقدمه:

همه کدهای زیر در یک فایل هستند. نام این فایل app.js است. یک کمپوننت از ان خارج میشود به نام App که در index.js میاد.

۲.۷.۱۱ قسمت صفر

```
import React, { Component } from 'react'
import { BrowserRouter, Route, Link} from 'react-router-dom'
```

۳.۷.۱۱ قسمت ۱

اولین کمپوننت که اخرین کمپوننتیه که در بروزر میاد. چون کدنویسی اینجا برعکسه! (چرا؟) چون در دل کمپوننت بعدی، کمپوننت قبلی قرار میگیره بخاطر همین معکوس میریم!

```
const Resource = ({ match }) => {

  const topic = topics.find(
    ({ id }) => id === match.params.topicId
  ).resources.find(
      ({ id }) => id === match.params.subId
    )

  return (
    <div>
      <h3>{topic.name}</h3>
      <p>{topic.description}</p>
      <a href={topic.url}>More info.</a>
    </div>
  )
}
```

۴.۷.۱۱ قسمت ۲

دومین کمپوننت که در مرحله یکی مونده به اخر در بروزر نمایش داده میشه. یعنی صفحه یکی مونده به اخری هست که ساختیمش.

```
const Topic = ({ match }) => {
  const topic = topics.find(
    ({ id }) => id === match.params.topicId
  )

  return (
    <BrowserRouter>
    <div>
      <h2>{topic.name}</h2>
      <p>{topic.description}</p>
      <ul>{topic.resources.map(
        (sub) => <li key={sub.id}>
              <Link to={`${match.url}/${sub.id}`}>{sub.name}</Link>
                </li>
        )}
        </ul><hr />
      <Route path={`${match.path}/:subId`} component={Resource}/>
    </div>
    </BrowserRouter>
  )
}
```

نکته: فکر کردم چون const داره قابل تعریف مجدد نیست اما در کد ها اینطور است. اگر ادرس دقیق میخوهید به تعریف کمپوننت های Resource ، Topic بروید!

۵.۷.۱۱ قسمت ۳

سومین کمپوننت.

```
const Topics = ({ match }) =>
<BrowserRouter>
<div><h1>Topics</h1>
<ul>{topics.map(
({ name, id }) =>
<li key={id}>
<Link to={`${match.url}/${id}`}>
{name}
</Link></li>
)}
</ul><hr />
<Route path={`${match.path}/:topicId`} component={Topic}/>
</div>
</BrowserRouter>
```

۶.۷.۱۱ قسمت ۴

```
const Home = () => <h1>Home</h1>
```

۷.۷.۱۱ قسمت ۵

```
export class App extends Component {
  render() {return (
      <BrowserRouter>
        <div style={{width: 1000, margin: '0 auto'}}>
          <ul>
            <li><Link to='/'>Home</Link></li>
            <li><Link to='/topics'>Topics</Link></li>
          </ul><hr />
          <Route exact path='/' component={Home} />
          <Route exact path='/topics' component={Topics} />
        </div>
      </BrowserRouter>
    )
  }
}
```

یه قسمت دیگه هم داره که مهم نیست فقط مقداردهیه!

۸.۷.۱۱ قسمت آخر:

```
const topics = [
  {
    name: 'React Router',
    id: 'react-router',
    description: 'Declarative, component based routing for React',
    resources: [
      {
        name: 'URL Parameters',
        id: 'url-parameters',
        description: "URL parameters are parameters whose values are set dynamically in a page's URL. This allows a route to render the same component while passing that component the dynamic portion of the URL so it can change based off of it.",
        url: 'https://tylermcginnis.com/react-router-url-parameters/'
      },
      {
        name: 'Programmatically navigate',
        id: 'programmatically-navigate',
        description: "When building an app with React Router, eventually you'll run into the question of navigating programmatically. The goal of this post is to break down the correct approaches to programmatically navigating with React Router.",
        url: 'https://tylermcginnis.com/react-router-programmatically-navigate/'
      }
    ]
  },
  {
    name: 'React.js',
    id: 'reactjs',
    description: 'A JavaScript library for building user interfaces',
    resources: [
      {
        name: 'React Lifecycle Events',
        id: 'react-lifecycle',
        description: "React Lifecycle events allow you to tie into specific phases of a components lifecycle",
        url: 'https://tylermcginnis.com/an-introduction-to-life-cycle-events-in-react-js/'
      },
      {
        name: 'React AHA Moments',
        id: 'react-aha',
        description: "A collection of 'Aha' moments while learning React.",
        url: 'https://tylermcginnis.com/react-aha-moments/'
      }
    ]
  },
  {
    name: 'Functional Programming',
    id: 'functional-programming',
    description: 'In computer science, functional programming is a programming -paradigma style of building the structure and elements of computer -programsthat treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data.',
    resources: [
      {
        name: 'Imperative vs Declarative programming',
        id: 'imperative-declarative',
```

```
        description: 'A guide to understanding the difference between Imperative and Declarative programming.',
        url: 'https://tylermcginnis.com/imperative-vs-declarative-programming/'
      },
      {
        name: 'Building User Interfaces with Pure Functions and Function Composition',
        id: 'fn-composition',
        description: 'A guide to building UI with pure functions and function composition in React',
        url: 'https://tylermcginnis.com/building-user-interfaces-with-pure-functions-and-function-composition-in-react-js/'
      }
    ]
  }
]
```

Reference:
https://tylermcginnis.com/react-router-nested-routes/

۸.۱۱ بیگ پروژه ۳: Nesting-Routes-Table

۱.۸.۱۱ قسمت ۰

```
import React,{Component} from 'react'
import {Link} from 'react-router-dom'
import {Switch,BrowserRouter,Route} from 'react-router-dom'
```

۲.۸.۱۱ قسمت ۱

```
export const Fn1 = ({key01,filter1}) => {
  const X1 = (!filter1 || !filter1.match(/key3|key2/)) ?
  key01 : key01.filter( x1 => x1[filter1] )
return (
<table>
  <thead>
    <tr>
      <th>Key1</th>
      <th>Key2</th>
      <th>Key3</th>
    </tr>
    <tr>
    <td colSpan={3}>
    <Link to='/mygo/all'>All</Link> ...
    <Link to='/mygo/key2'>Key2</Link> ...
    <Link to='/mygo/key3'>Key3</Link>
    </td>
    </tr>
  </thead>
  <tbody>
    {X1.map(Fn2)}
  </tbody>
</table>
)}
```

۳.۸.۱۱ قسمت ۲

```
const Fn2 = ({key1,key2,key3}) =>
<tr>
<td>{key1}</td>
<td>{key2 ? "lala" : null}</td>
<td>{key3 ? "good" : null}</td>
</tr>
```

۴.۸.۱۱ قسمت ۳

```
export class App extends Component{
  render(){
    return(
    <div><h1>Hello!</h1>
    <BrowserRouter>
    <div>
      <Link to={`${this.props.match.url}/key3`}> Key3 </Link>
      <hr/>
      <Route exact path={`${this.props.match.path}/:x`} render={
({match}) => <Fn1 key01={value01} filter1={match.params.x}/>
      }/>
    </div>
    </BrowserRouter>
    </div>
  )}
}
```

۵.۸.۱۱ قسمت ۴

```
var value01 = [
  {
    key1: "value11",
    key2: false,
    key3: false,
  },
  {
    key1: "value21",
    key2: false,
    key3: false,
  },
  {
    key1: "value31",
    key2: true,
    key3: false,
  },
  {
    key1: "value41",
    key2: true,
    key3: false,
  },
  {
    key1: "value51",
    key2: false,
    key3: true,
  },
]
```

۶.۸.۱۱ قسمت ۵

```
export const Err = () =>
<div>
<h1>Error 404!</h1>
Page Not Found!
<button><Link to="/">Back Home</Link></button>
</div>
```

۷.۸.۱۱ قسمت ۶: برگ index.js

```
import React from 'react'
import { render } from 'react-dom'
import { Sq,Err,App1,App} from './app'
import {Switch,BrowserRouter,Route} from 'react-router-dom'
window.React = React

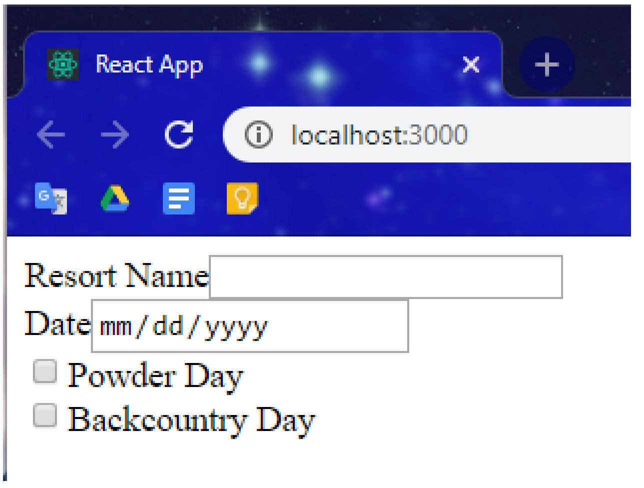
render(
  <BrowserRouter>
  <Switch>
  <Route exact path="/mygo" component={App} />
  <Route component={Err} />
  </Switch>
  </BrowserRouter>
  ,document.getElementById('root')
)
```

فصل ۱۲

افزودن AddDay

😲 در این فصل یک پروژه داریم که حاصل پنج قسمت بزرگ است.

۱.۱۲ ایجاد یک form



ظاهر فرم را میسازیم با jsx .

ref: https://reactjs.org/docs/uncontrolled-components.html

راهنما در لینک بالا

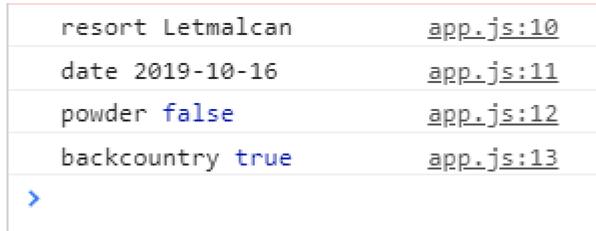
```
import React,{Component} from 'react'
import PropTypes from 'prop-types'
import {BrowserRouter,Route,Link} from 'react-router-dom'

export class AddDayForm extends Component {
  render(){
const {resort,date,powder,backcountry} = this.props
    return(
<div>
<form>

<label htmlFor="resort">Resort Name</label>
<input id="resort" type="text" required defaultValue={resort} />
<br/>
<label htmlFor="date">Date</label>
<input id="date" type="date" required defaultValue={date} />
<br/>
<div>
<input id="powder" type="checkbox" defaultChecked={powder} />
<label htmlFor="powder">Powder Day</label>
</div>
<div>
<input id="backcountry" type="checkbox" defaultChecked={backcountry} />
<label htmlFor="backcountry">Backcountry Day</label>
</div>


</form>
</div>)
  }
}
AddDayForm.propTypes = {
  resort: PropTypes.string.isRequired,
  date: PropTypes.string.isRequired,
  powder: PropTypes.bool.isRequired,
  backcountry: PropTypes.bool.isRequired
}
```

۲.۱۲ دریافت از form



برای اینکه ببینیم مقدار میگیره یا نه میگیم در کنسول نشون بده!

```
import React,{Component} from 'react'
import PropTypes from 'prop-types'

export class AddDayForm extends Component {
  constructor(props){ super(props)
    this.submit = this.submit.bind(this)
  }

  submit(e) { e.preventDefault()
    console.log('resort',this.refs.resort.value)
    console.log('date',this.refs.date.value)
    console.log('powder',this.refs.powder.checked)
    console.log('backcountry',this.refs.backcountry.checked)
  }

  render(){
const {resort۱,date۱,powder۱,backcountry۱} = this.props

    return(
<div>
<form onSubmit={this.submit}>

<label htmlFor="resort">Resort Name</label>
<input id="resort" type="text" defaultValue={resort۱} required
    ref="resort" />
<br/>
<label htmlFor="date">Date</label>
<input id="date" type="date"  defaultValue={date۱} required ref=
    "date"/>
<br/>
<div>
<input id="powder" type="checkbox" defaultChecked={powder۱} ref=
    "powder"/>
<label htmlFor="powder">Powder Day</label>
</div>
<div>
<input  id="backcountry" type="checkbox" defaultChecked={
    backcountry۱} ref="backcountry"/>
<label htmlFor="backcountry">Backcountry Day</label>
</div>

<button>Add Form</button>
</form>
</div>)
  }
}

AddDayForm.defaultProps={
  resort۱: "Letmalcan",
  date۱: "98.07.10",
  powder۱: false,
  backcountry۱: true
}

AddDayForm.propTypes = {
  resort: PropTypes.string.isRequired,
```

```
  date: PropTypes.string.isRequired,
  powder: PropTypes.bool.isRequired,
  backcountry: PropTypes.bool.isRequired
}
```

۳.۱۲ تبدیل کلاس به تابع

اینجا کمپوننت فرم که یک کلاس بوده به تابع تبدیل میشه و کدهای فرم خونه نو میخرن 😃

```
import React,{Component} from 'react'
import PropTypes from 'prop-types'


export const AddDayForm = ({resort۱,date۱,powder۱,backcountry۱})
    => {
  const submit = (e) => {
    e.preventDefault()
    console.log('resort', _resort.value)
    console.log('date', _date.value)
    console.log('powder', _powder.checked)
    console.log('backcountry', _backcountry.checked)
  }
  let _resort, _date, _backcountry, _powder
  return(
  <div>
  <form onSubmit={submit}>

  <label htmlFor="resort">Resort Name</label>
  <input id="resort" type="text" defaultValue={resort۱} required
       ref={input => _resort = input} />
  <br/>
  <label htmlFor="date">Date</label>
  <input id="date" type="text"  defaultValue={date۱} required
      ref={input => _date = input}/>
  <br/>
  <div>
  <input id="powder" type="checkbox" defaultChecked={powder۱}
      ref={input => _powder = input}/>
  <label htmlFor="powder">Powder Day</label>
  </div>
  <div>
  <input  id="backcountry" type="checkbox" defaultChecked={
      backcountry۱} ref={input => _backcountry = input}/>
  <label htmlFor="backcountry">Backcountry Day</label>
  </div>

  <button>Add Form</button>
  </form>
  </div>)

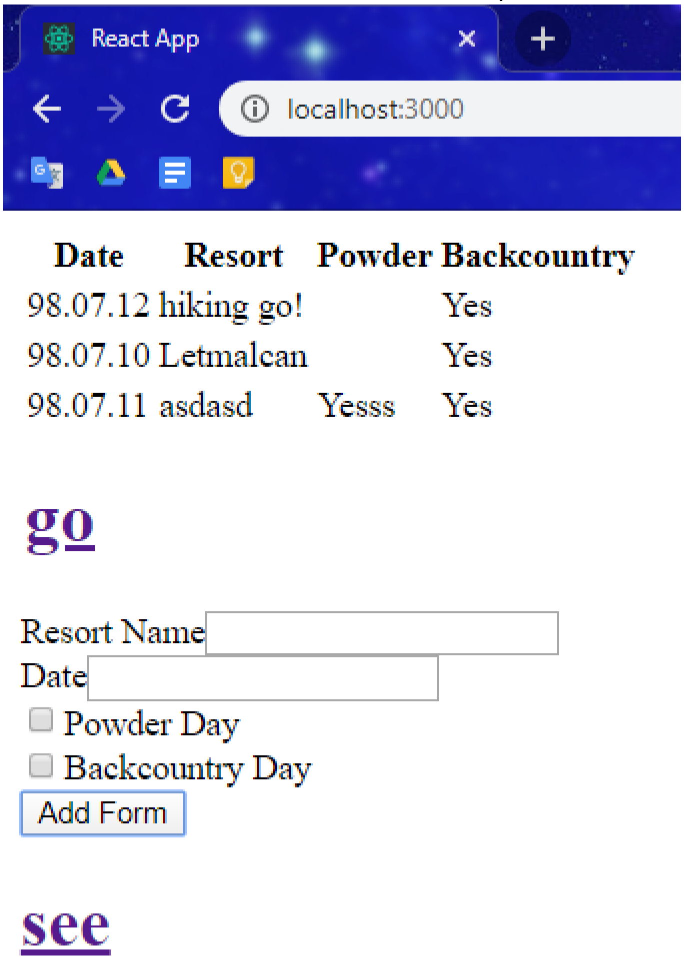

}

AddDayForm.defaultProps={
  resort۱: "Letmalcan",
  date۱: "98.07.10",
  powder۱: false,
  backcountry۱: true
}

AddDayForm.propTypes = {
  resort: PropTypes.string.isRequired,
  date: PropTypes.string.isRequired,
  powder: PropTypes.bool,
  backcountry: PropTypes.bool
}
```

۴.۱۲ اضافه کردن مقدار جدول

این بخش از بقیه لذتش بیشتره! قراره مقدار به جدول اضافه کنیم. خوشم اومد. منم میتونم این اپلیکیشن را برای خودم درست کنم. تا خیلی سریع اطلاعات مکانهایی که میرمو در یک جدول با یک کلیک اضاف کنم.



setState: https://reactjs.org/docs/react-component.html#setstate

```
import React,{Component} from 'react'
import {Link} from 'react-router-dom'
import PropTypes from 'prop-types'


const AddDayForm = ({resort,date,powder,backcountry,x2}) => {
  const submit = (e) => {
    e.preventDefault()
    x2({
      resort: _resort.value,
      date: _date.value,
      powder: _powder.checked,
      backcountry: _backcountry.checked,
    })
     _resort.value=''
     _date.value=''
     _powder.checked=false
     _backcountry.checked=false
  }
  let _resort, _date, _backcountry, _powder
  return(
  <div>

  <form onSubmit={submit}>

  <label htmlFor="resort">Resort Name</label>
```

```
  <input id="resort" type="text" defaultValue={resort} required
      ref={x => _resort = x} />
  <br/>
  <label htmlFor="date">Date</label>
  <input id="date" type="text"  defaultValue={date} required ref
      ={x => _date = x}/>
  <br/>
  <div>
  <input id="powder" type="checkbox" defaultChecked={powder} ref
      ={x => _powder = x}/>
  <label htmlFor="powder">Powder Day</label>
  </div>
  <div>
  <input  id="backcountry" type="checkbox" defaultChecked={
      backcountry} ref={x => _backcountry = x}/>
  <label htmlFor="backcountry">Backcountry Day</label>
  </div>

  <button>Add Form</button>
  </form>
    <h1><Link to="/">see</Link></h1>
  </div>)


}

AddDayForm.defaultProps={
  resort: "Letmalcan",
  date: "98.07.10",
  powder: false,
  backcountry: true
}

AddDayForm.propTypes = {
  resort: PropTypes.string.isRequired,
  date: PropTypes.string.isRequired,
  powder: PropTypes.bool,
  backcountry: PropTypes.bool
}

const Fn1 = ({key01}) =>
<table>
<thead>
<tr>
<th>Date</th>
<th>Resort</th>
<th>Powder</th>
<th>Backcountry</th>
</tr>
</thead>
<tbody>
{key01.map(Fn2)}
</tbody>
<h1><Link to="/go">go</Link></h1>
</table>

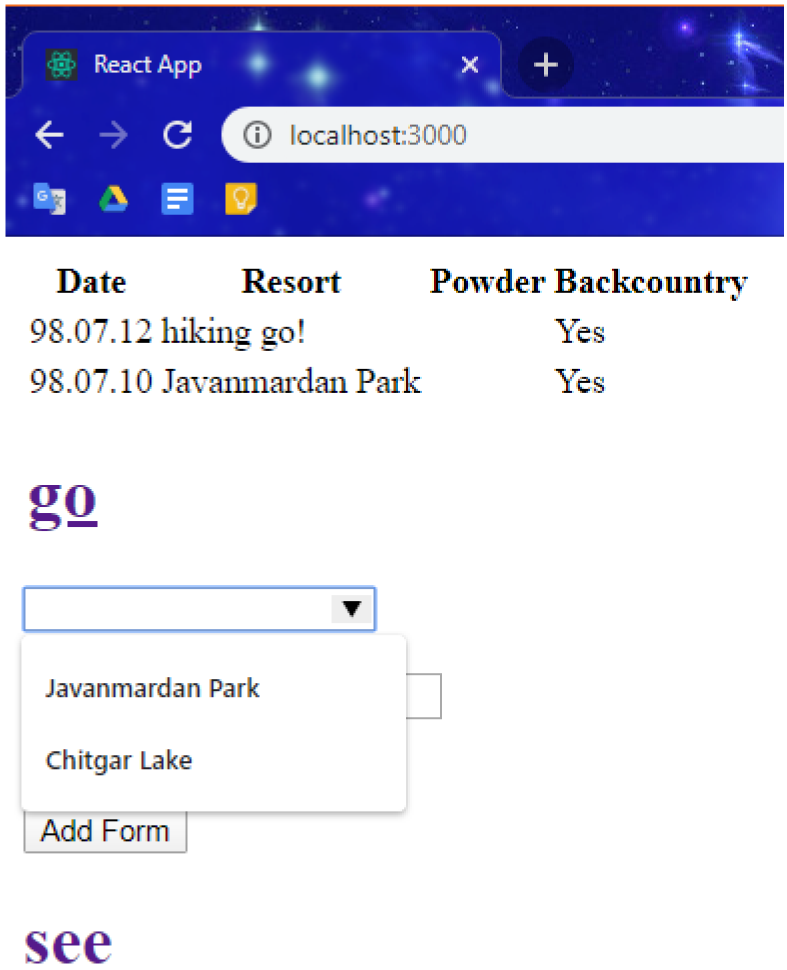
const Fn2 = ({resort,date,powder,backcountry}) =>
<tr>
<td>{date}</td>
<td>{resort}</td>
<td>{powder ? "Yesss" : null }</td>
<td>{backcountry ? "Yes" : null }</td>
</tr>

export class App extends Component{
  constructor(props){super(props)
    this.state = {
      value01: [{
        resort: "hiking go!",
```

```
          date: "98.07.12",
          powder: false,
          backcountry: true,
        },
      ]}
      this.addDay = this.addDay.bind(this)
    }
    addDay(x1){
      this.setState({
        value01:[
          ...this.state.value01, x1 ] }
        )

    }
      render(){return(<div>
              <Fn1 key01={this.state.value01}/>
              <AddDayForm x2={this.addDay} />
            </div>)
      }
}
```

۵.۱۲ اپشن option




```
import React,{Component} from 'react'
import {Link} from 'react-router-dom'
import PropTypes from 'prop-types'

const x3 = [
  'Javanmardan Park',
  'Chitgar Lake'
]
class Autocomplete extends Component{
  get value(){ return this.refs.x11.value}
  set value(xy){this.refs.x11.value = xy}

  render(){return <div>
    <input type='text' list='x-x' ref ='x11'/>
    <datalist id='x-x'>
      {this.props.options.map(
(opt,i) => <option key={i}> {opt} </option>
      )}
    </datalist>
    </div>

  }
}
const AddDayForm = ({resort,date,powder,backcountry,x2}) => {
  const submit = (e) => {
    e.preventDefault()
    x2({
      resort: _resort.value,
      date: _date.value,
      powder: _powder.checked,
      backcountry: _backcountry.checked,
    })
```

```
      _resort.value=''
      _date.value=''
      _powder.checked=false
      _backcountry.checked=false
  }
  let _resort, _date, _backcountry, _powder
  return(
  <div>

  <form onSubmit={submit}>
  <Autocomplete options={x3} ref={x => _resort = x} />
  <br/>
  <label htmlFor="date">Date</label>
  <input id="date" type="text"  defaultValue={date} required ref
      ={x => _date = x}/>
  <br/>
  <div>
  <input id="powder" type="checkbox" defaultChecked={powder} ref
      ={x => _powder = x}/>
  <label htmlFor="powder">Powder Day</label>
  </div>
  <div>
  <input  id="backcountry" type="checkbox" defaultChecked={
      backcountry} ref={x => _backcountry = x}/>
  <label htmlFor="backcountry">Backcountry Day</label>
  </div>

  <button>Add Form</button>
  </form>
    <h1><Link to="/">see</Link></h1>
  </div>)


}

AddDayForm.defaultProps={
  resort: "Letmalcan",
  date: "98.07.10",
  powder: false,
  backcountry: true
}

AddDayForm.propTypes = {
  resort: PropTypes.string.isRequired,
  date: PropTypes.string.isRequired,
  powder: PropTypes.bool,
  backcountry: PropTypes.bool
}

const Fn1 = ({key01}) =>
<table>
<thead>
<tr>
<th>Date</th>
<th>Resort</th>
<th>Powder</th>
<th>Backcountry</th>
</tr>
</thead>
<tbody>
{key01.map(Fn2)}
</tbody>
<h1><Link to="/go">go</Link></h1>
</table>

const Fn2 = ({resort,date,powder,backcountry}) =>
<tr>
<td>{date}</td>
<td>{resort}</td>
```

```
<td>{powder ? "Yesss" : null }</td>
<td>{backcountry ? "Yes" : null }</td>
</tr>

export class App extends Component{
  constructor(props){super(props)
    this.state = {
      value01: [{
        resort: "hiking go!",
        date: "98.07.12",
        powder: false,
        backcountry: true,
      },
    ]}
    this.addDay = this.addDay.bind(this)
  }
  addDay(x1){
    this.setState({
      value01:[
        ...this.state.value01, x1 ] }
      )

  }
    render(){return(<div>
          <Fn1 key01={this.state.value01}/>
          <AddDayForm x2={this.addDay} />
        </div>)
    }
}
```

فصل ۱۳

اپلیکیشن دوم: نمایش اعضا

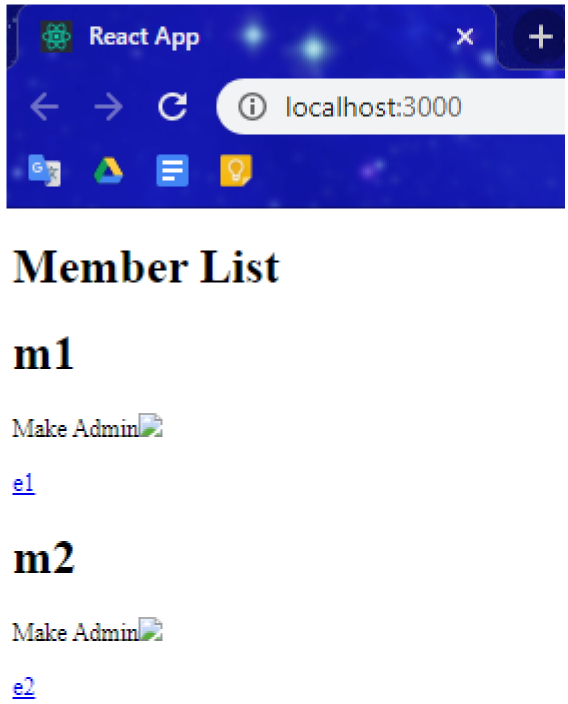
😀 در این فصل که اخرین فصل است. میتوانید به کدهای اخرین بخش که عکس یادگاری هستند بروید. همچنین میتوانید به ترتیب پیش بروید و مسیر فکری داشته باشید!

۱.۱۳ ساختن کمپوننت عضو

```
import React ,{Component} from 'react'

export class Member extends Component{
  makeAdmin(x){console.log(x)}
  render(){
    const {name,email,thumbnail,admin} = this.props
    return(
      <div>
      <h1>{name} {admin ? "icon" : null}</h1>
      <a onClick={this.makeAdmin}>Make Admin</a>
      <img src={thumbnail}/>
      <p><a href={`mailto:${email}`}> {email} </a></p>
      </div>
  )}
}
```

۲.۱۳ ساختن کمپوننت لیست

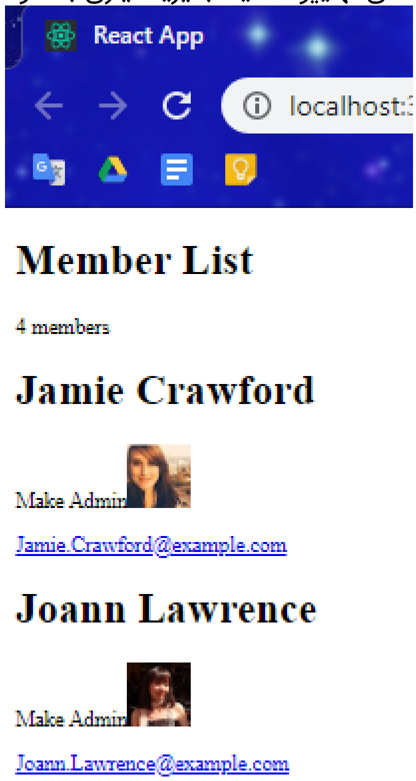



```
export class Memberlist extends Component {
  constructor(props){super(props)
this.state = {
  members: [
    {
      name: 'm1',
      email: 'e1',
      thumbnail: 'img1.png'
    },{
      name: 'm2',
      email: 'e2',
      thumbnail: 'img2.png'
    },{
      name: 'm3',
      email: 'e3',
      thumbnail: 'img3.png'
    },{
      name: 'm4',
      email: 'e4',
      thumbnail: 'img4.png'
    },

  ]
}
  }
  render(){
    const {members} = this.state
    return(
      <div>
      <h1>Member List</h1>
      {members.map(
        (x,i) => <Member key={i}
                   onClick={x => console.log(x)}
                  {...x}/>)
                }
      </div>
    )
  }
}
```

۳.۱۳ گرفتن اعضا از اینترنت با fetch

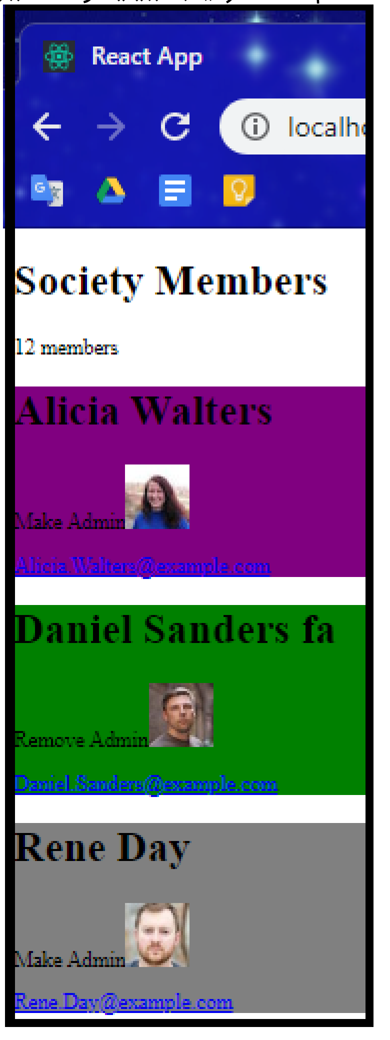
ممکن است بخشی از این کدها (بخاطر اینکه خودم نوشتمشون) با کدهای نهایی فرق کنه! شما کدهای نهاییو اگه یاد بگیرید نیازی به خواندن این کدها ندارید!




```
export class Memberlist extends Component {
  constructor(props){super(props)
this.state = {
  members: [],
  loading: false
}
  }
  componentDidMount(){
    this.setState({loading: true})
    fetch("https://api.randomuser.me/?nat=US&results=4")
    .then(response => response.json())
    .then(json => json.results)
    .then(members => this.setState({
      members,
      loading: false
    }))
  }
  render(){
    const {members,loading} = this.state
    return(
      <div>
      <h1>Member List</h1>
      {loading ? <span>loading...</span> : <span>{members.length} members</span>}
      {members.length ?
        members.map(
        (member,i) => <Member key={i}
                          name={member.name.first + ' ' + member.name.last}
                          email={member.email}
                          thumbnail={member.picture.thumbnail} />) :
        <span>currently 0 members</span>
                    }
      </div>
    )
  } }
```

۴.۱۳ عکس یادگاری

😍 تمام کدها در اینجا ببینید و لذت ببرید!




```
import React from 'react'
import { Component } from 'react'
```

اولین کمپوننت را در صفحه بعد ببینید!

۱.۴.۱۳ کمپوننت Member

```
class Member extends Component {

  componentWillMount() {
    this.style = {
      backgroundColor: 'gray'
    }
  }

  componentWillUpdate(nextProps) {
    this.style = {
      backgroundColor: (nextProps.admin) ? 'green' : 'purple'
    }
  }

  shouldComponentUpdate(nextProps) {
    return this.props.admin !== nextProps.admin
  }

  render() {

  const { name, thumbnail, email, admin,
          makeAdmin, removeAdmin } = this.props
    return (
        <div style={this.style}>
          <h1>{name} {(admin) ? "fa" : null}</h1>
          {(admin) ?
            <a onClick={() => removeAdmin(email)}>Remove Admin</a> :
            <a onClick={() => makeAdmin(email)}>Make Admin</a>
          }


          <img src={thumbnail} />
          <p><a href={`mailto:${email}`}>{email}</a></p>

        </div>
    )
}
}
```

در صفحه بعد از تماشای عکس یادگاری کدهای تشکیل دهنده کمپوننت MemberList لذت ببرید!

۲.۴.۱۳ کمپوننت MemberList

این شما و این اخرین قطعه کد از فصل لیندا! ❤️ خیلی خوش گذشت!! محال یادم بره! خاطرات لیندا ری اکت!

```
class MemberList extends Component {

    constructor(props) {
        super(props)
        this.state = {
            members: [],
            loading: false,
            administrators: []
        }
        this.makeAdmin = this.makeAdmin.bind(this)
        this.removeAdmin = this.removeAdmin.bind(this)
    }

        componentWillUpdate(nextProps) {
        this.style = { backgroundColor: (nextProps.admin) ? '
            green' : 'purple' }
    }

    componentDidUpdate(prevProps) {
       console.log(`${prevProps.name} updated`, prevProps.admin,
            this.props.admin)
    }

    componentDidMount() {
        this.setState({loading: true})
        fetch('https://api.randomuser.me/?nat=US&results=12')
            .then(response => response.json())
            .then(json => json.results)
            .then(members => this.setState({
                members,
                loading: false
            }))
    }

    makeAdmin(email) {
        const administrators = [
            ...this.state.administrators,
            email
        ]
        this.setState({administrators})
    }

    removeAdmin(email) {
        const administrators = this.state.administrators.filter(
            adminEmail => adminEmail !== email
        )
        this.setState({administrators})
    }


    render() {
      const { administrators, members, loading } = this.state
        return (
            <div className="member-list">
                <h1>Society Members</h1>

                {(loading) ?
                    <span>loading...</span> :
                    <span>{members.length} members</span>
                }

                {(members.length) ?
                   members.map(
                  (member, i) =>
```

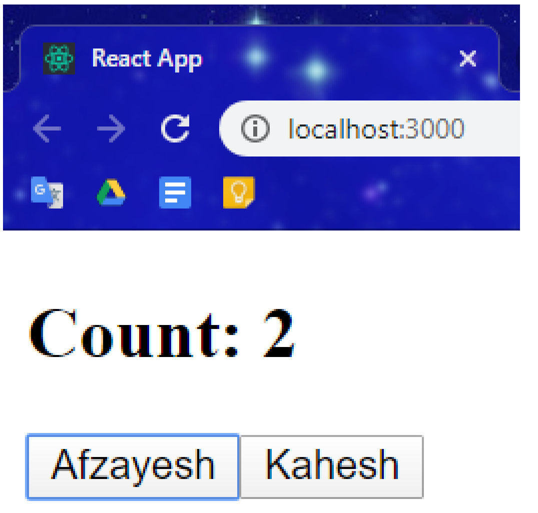
```
                    <Member key={i}
                            admin={administrators.some(
                                adminEmail => adminEmail === member.email
                                )}
                            name={member.name.first + ' ' + member.name.last}
                            email={member.email}
                            thumbnail={member.picture.thumbnail}
                            makeAdmin={this.makeAdmin}
                            removeAdmin={this.removeAdmin}/>
                    ) :
                    <span>Currently o Members </span>
                }
            </div>
        )
    }
}

export default MemberList
```

فصل ۱۴

سلام Redux عزیز!

۱.۱۴ اولین Redux من با استاد صدری!



۱.۱.۱۴ شمارنده با React

پیشنیاز

```
npx install create-react-app
```

تمام کدها (عکس یادگاری)

```
import React,{Component} from 'react'

export class App extends Component{
  constructor(props){super(props)
    this.fn1 = this.fn1.bind(this)
    this.fn2 = this.fn2.bind(this)
    this.state = {count:0}
  }

  fn1(){this.setState( {count: this.state.count + 1} ) }
  fn2(){this.setState( {count: this.state.count-1} ) }
  render(){return(
    <div>
    <h2>Count: {this.state.count}</h2>
    <button onClick={this.fn1}>Afzayesh</button>
    <button onClick={this.fn2}>Kahesh</button>
    </div>
  ) } }
```

۲.۱.۱۴ همون شمارنده با Redux

😀 پیشنیاز

```
npm install redux --save
```

یه سایت همینجوری پیدا کردم.

https://redux-actions.js.org/api/createaction

۱.۲.۱.۱۴ فایل components/counter.js

```
import React,{Component} from 'react'

export class Counter extends Component{
render(){
  const {value,onIncrement,onDecrement} = this.props
  return(
    <div>
    <h2>Counter: {value}</h2>
    <button onClick={onIncrement}>Increment</button>
    <button onClick={onDecrement}>Decrement</button>
    </div>
  ) } }
```

۲.۲.۱.۱۴ فایل reducers/index.js

```
export default (state=0 , action) => {
  switch(action.type){
    case 'INCREMENT': return state+1
    case 'DECREMENT': return state-1
    default: return state
  } }
```

۳.۲.۱.۱۴ فایل index.js

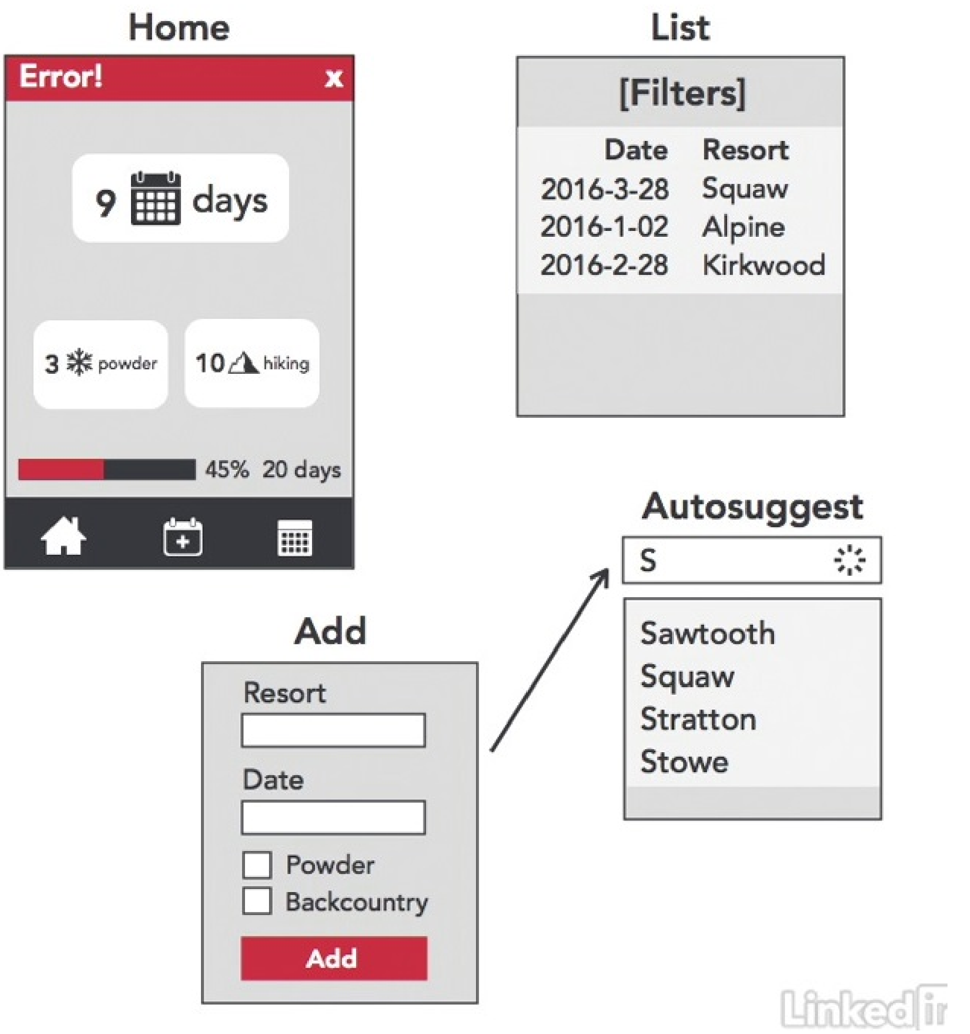
```
import React from 'react'
import { render } from 'react-dom'
import { createStore } from 'redux'
import {Counter} from './components/counter'
import counter from './reducers'

const store = createStore(counter)
const render1 = () => render(
  <Counter value={store.getState()}
  onIncrement={() => store.dispatch({type:'INCREMENT'})}
  onDecrement={() => store.dispatch({type:'DECREMENT'})}
  />, document.querySelector("#root"))
render1()
store.subscribe(render1)
```

۲.۱۴ آخ جون Plan-a-Redux-App

از اینجا به بعد دومین پروژه شروع میشه! تا اخر این کتاب همراهمونه! در طول این دوره از اساس reducer و store و action بهمون گفته میشه. کلی لذت میبریم!

۱.۲.۱۴ وایرفریم wireframe میکشیم!



۲.۲.۱۴ ابجکت Object های بکار رفته:

- View
- Model
- Collection
- skiDayCount
- skiDaysCollection
- NavMenuView
- skiDayModel
- RouteController
- AutoSuggest

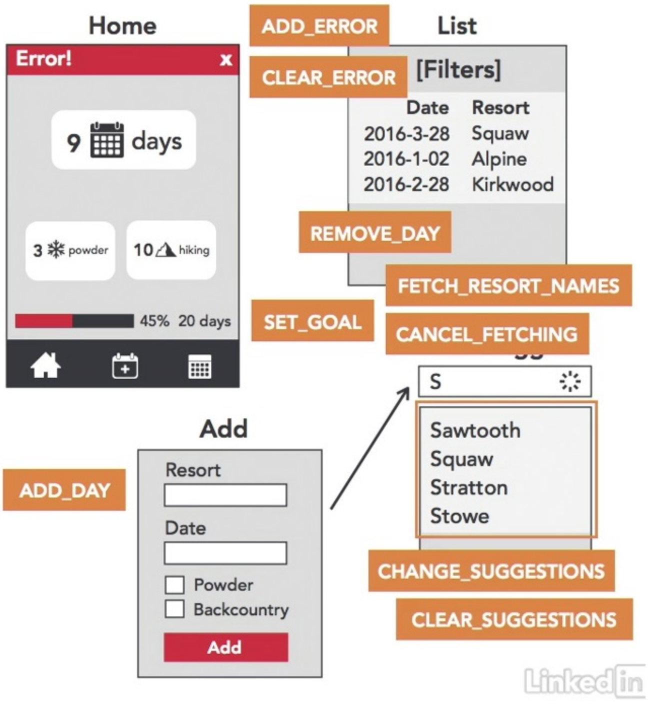
۳.۲.۱۴ اکشن action هایی که در این پروژه استفاده کردیم:

- ADD_DAY
- REMOVE_DAY
- SET_GOAL
- ADD_ERROR
- CLEAR_ERROR
- FETCH_RESORT_NAMES
- CANCEL_FETCHING
- CHANGE_SUGGESTIONS
- CLEAR_SUGGESTIONS

constants.js

```
const constants = {
  ADD_DAY: "ADD_DAY",
  REMOVE_DAY: "REMOVE_DAY",
  SET_GOAL: "SET_GOAL",
  ADD_ERROR: "ADD_ERROR",
  CLEAR_ERROR: "CLEAR_ERROR",
  FETCH_RESORT_NAMES: "FETCH_RESORT_NAMES",
  CANCEL_FETCHING: "CANCEL_FETCHING",
  CHANGE_SUGGESTIONS: "CHANGE_SUGGESTIONS",
  CLEAR_SUGGESTIONS: "CLEAR_SUGGESTIONS"
}


export default constants
```



۴.۲.۱۴ استیت state هایی که استفاده شد:

- skiDays -> object
- allSkiDays -> array
- goal -> number
- errors -> array
- resortNames.fetching -> boolean
- resortNames.suggestions -> array




```
{
    "allSkiDays": [
        {
            "resort": "Kirkwood",
            "date": "2016-12-7",
            "powder": true,
            "backcountry": false
        },
        {
            "resort": "Squaw Valley",
            "date": "2016-12-8",
            "powder": false,
            "backcountry": false
        },
        {
            "resort": "Mt Tallac",
            "date": "2016-12-9",
            "powder": false,
            "backcountry": true
        }
    ],
    "goal": 10,
    "errors": [],
    "resortNames": {
        "fetching": false,
        "suggestions": ["Squaw Valley", "Snowbird", "Stowe", "Steamboat"]
    }
}
```

۳.۱۴ راه اندازی babel

میخوایم با بابل ترنسپیل کنیم! :) یادته محمدامین؟
تمام محتویات داخل پوشه rx-html پاک کردم و از اول شروع کردم.

۱.۳.۱۴ ایجاد فایل package.json

```
npm init
```

۲.۳.۱۴ نصب پکیج ها

```
npm install babel-cli --save
npm install babel-preset-latest --save
npm install babel-preset-stage-o --save
```

۳.۳.۱۴ پوشه بندی

- Rx-html
  - src
    - index.js
    - initialState.json
    - constants.js
  - .babelrc
  - package.json

۴.۳.۱۴ ایجاد فایل babelrc

نکته در نامگذاری فایل این است که فقط پسوند دارد و نام ندارد.

```
// .babelrc
{ "presets" : ["latest","stage-0"] }
```

۵.۳.۱۴ تغییر فایل package.json

```
"scripts": {
  "start": "./node_modules/.bin/babel-node ./src",
```

۶.۳.۱۴ ایجاد فایل constants.js

```
const constants = {
  ADD_DAY: "ADD_DAY",
  REMOVE_DAY: "REMOVE_DAY",
  SET_GOAL: "SET_GOAL",
  ADD_ERROR:"ADD_ERROR",
  CLEAR_ERROR:"CLEAR_ERROR",
  FETCH_RESORT_NAMES:"FETCH_RESORT_NAMES",
  CANCEL_FETCHING: "CANCEL_FETCHING",
  CHANGE_SUGGESTIONS:"CHANGE_SUGGESTIONS",
  CLEAR_SUGGESTIONS: "CLEAR_SUGGESTIONS",
}

export default constants
```

۷.۳.۱۴ ایجاد فایل initialState.json

```
{
  "allSkiDays":[
    {
      "resort": "Letmalcan Park",
      "date": "2016-12-7",
      "powder": true,
      "backcountry": false
    },{
      "resort": "Javanmardan Park",
      "date": "2016-12-8",
      "powder": false,
      "backcountry": false
    },{
      "resort": "Mountains",
      "date": "2016-12-9",
      "powder": true,
      "backcountry": false
    }
  ],
  "goal": 10,
  "error":[],
  "resortNames":{
    "fetching": false,
    "suggestions":["Chitgar pist","Chitgar Lake","Letmalcan
        Mountains", "Kolakchal Mountains"]
  }
}
```

۸.۳.۱۴ ایجاد فایل index.js

```
import C from './constants'
import { allSkiDays, goal } from './initialState.json'

console.log(`

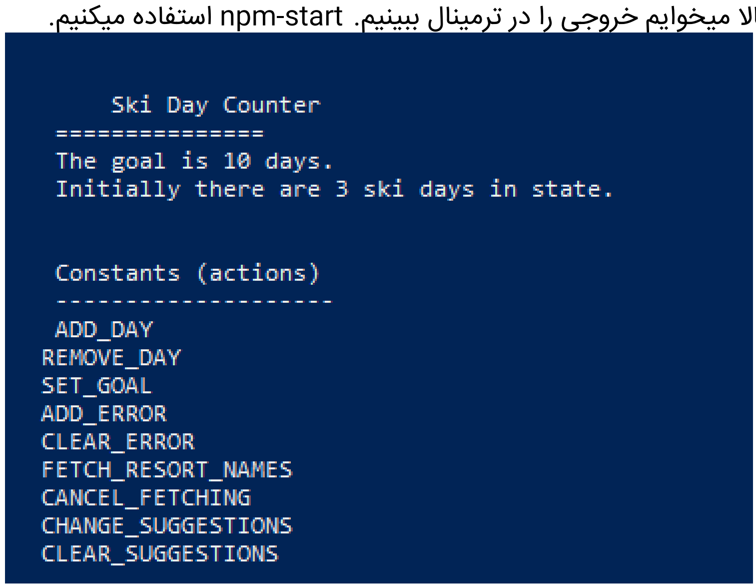

  Ski Day Counter
    ===============
    The goal is ${goal} days.
    Initially there are ${allSkiDays.length} ski days in state.


    Constants (actions)
    --------------------
    ${Object.keys(C).join('\n    ')}

  `)
```

۹.۳.۱۴ خروجی

حالا میخوایم خروجی را در ترمینال ببینیم. npm-start استفاده میکنیم.



یکبار برام ارور داد! اگه در فایل constants خروجی export را همراه با default در پایین تعریف ابجکت اورده باشیم ارور نمیدهد و خروجی این شکلی میشود.

۴.۱۴ اولین Reducer من :)

۱.۴.۱۴ فایل reducers.js

در پوشه src یک پوشه دیگر درست میکنیم اسمش را میگذاریم store یه فایل به نام reducers.js در ان ایجاد میکنیم و در ان مینویسیم:

```
import C from '../constants'

export const goal = (state=10 , action) => {
  if(action.type === C.SET_GOAL){
    return parseInt(action.payload)
  }else{ return state }
}
```

چون رنگ بنفش زیاد توش داشت خوشم اومد گفتم عکسشم بذارم ببینی محمدامین! ببین چقد قشنگه!

```
import C from '../constants'

export const goal = (state=10 , action) => {
  if(action.type === C.SET_GOAL){
    return parseInt(action.payload)
  }else{ return state }
}
```

۲.۴.۱۴ فایل index.js

در فایل index.js در src هم این را مینویسیم:

```
import C from './constants'
import { goal } from './store/reducers'

const state = 10

const action = {
  type: C.SET_GOAL,
  payload: 15
}

const nextState = goal(state,action)

console.log(`

      initial goal: ${state}
      action: ${JSON.stringify(action)}
      new goal: ${nextState}

  `)
```

۳.۴.۱۴ خروجی

حالا npm-start استفاده میکنیم (در مسیر پوشه Desktop/Rx-html)

۵.۱۴ معرفی object-reducer

۱.۵.۱۴ فایل reducers.js

این قطعه کد اضافه میشود:

```
export const skiDay = (state=null, action) =>
(action.type === C.ADD_DAY) ? action.payload : state
```

۲.۵.۱۴ فایل index.js

حالا همون قطعه کد (object-reducer) را اینجا فراخوانی میکنیم:

```
import C from './constants'
import { skiDay } from './store/reducers'

const state = null

const action = {
  type: C.ADD_DAY,
  payload: {
    "resort": "Letmalcan Park",
    "date": "2016-12-16",
    "powder": true,
    "backcountry": false
  }
}

const nextState = skiDay(state, action)

console.log(`

      initial state: ${state}
      action: ${JSON.stringify(action)}
      new state: ${JSON.stringify(nextState)}

  `)
```

۳.۵.۱۴ خروجی

حالا npm-start استفاده میکنیم (در مسیر پوشه Desktop/Rx-html)

۶.۱۴ معرفی array-reducer

۱.۶.۱۴ فایل reducers.js

این قطعه کد اضافه میشود:

```
export const errors = (state=[],action) => {
  switch(action.type){
    case C.ADD_ERROR: return [...state,action.payload]
    default: return state
  }
}
```

۲.۶.۱۴ فایل index.js

حالا اینجا فراخوانیش میکنیم:

```
import C from './constants'
import { errors } from './store/reducers'

const state = [ "user not authorized", "server feed not found" ]

const action = {
  type: C.ADD_ERROR,
  payload: 'cannot connect to server'
}

const nextState = errors(state, action)

console.log(`

      initial state: ${state}
      action: ${JSON.stringify(action)}
      new state: ${JSON.stringify(nextState)}

  `)
```

۳.۶.۱۴ خروجی

حالا npm-start استفاده میکنیم (در مسیر پوشه Desktop/Rx-html)
این داستان ادامه دارد ... (در صفحه بعد)

در ادامه صفحه قبل اینجا اضافه میشه:

۴.۶.۱۴ فایل reducers.js

اکشن پاک کردن ارور اضافه میشود:

```
export const errors = (state=[],action) => {
  switch(action.type){
    case C.ADD_ERROR: return [...state,action.payload]
    case C.CLEAR_ERROR: return state.filter(
      (message,i) => i !== action.payload
    )
    default: return state
  }
}
```

۵.۶.۱۴ فایل index.js

فراخوانیش میکنیم:

```
import C from './constants'
import { errors } from './store/reducers'

const state = [ "user not authorized", "server feed not found" ]

const action = {
  type: C.CLEAR_ERROR,
  payload: 0
}

const nextState = errors(state, action)

console.log(`

      initial state: ${state}
      action: ${JSON.stringify(action)}
      new state: ${JSON.stringify(nextState)}

  `)
```

۶.۶.۱۴ خروجی

حالا npm-start استفاده میکنیم

۷.۱۴ نوشتن reducer : شماره ۱

۱.۷.۱۴ فایل reducers.js

این تکه اضافه میشود:

```
export const allSkiDays = (state=[],action) => {
  switch(action.type){
    case C.ADD_DAY: return [...state,skiDay(null,action)]
    default: state
  } }
```

۲.۷.۱۴ فایل index.js

```
import C from './constants'
import { allSkiDays } from './store/reducers'

const state = [{
  "resort": "Letmalcan Park",
  "date": "2016-12-6",
  "powder": true,
  "backcountry": false
}]
const action = {
  type: C.ADD_DAY,
  payload: {
    "resort": "Chitgar Park",
    "date": "2016-12-7",
    "powder": true,
    "backcountry": false
  }
}

const nextState = allSkiDays(state, action)

console.log(`

      initial state: ${JSON.stringify(state)}
      action: ${JSON.stringify(action)}
      new state: ${JSON.stringify(nextState)}

  `)
```

۳.۷.۱۴ خروجی

حالا npm-start استفاده میکنیم

۸.۱۴ نوشتن reducer : شماره ۲

۱.۸.۱۴ فایل reducers.js

😍 فقط قطعه کد های allSkiDays عوض میشه حتی index.js هم تغییر نمیکنه و یراست میریم ترمینال برای خروجی!

```
export const allSkiDays = (state=[],action) => {
  switch(action.type){
    case C.ADD_DAY:
    const hasDay = state.some(
      skiDay => skiDay.date === action.payload.date)
      return (hasDay) ? state: [...state,skiDay(null,action)]
    default: state
  }
}
```

۲.۸.۱۴ خروجی

حالا npm-start استفاده میکنیم

۹.۱۴ نوشتن reducer : شماره ۳

۱.۹.۱۴ فایل reducers.js

```
export const allSkiDays = (state=[],action) => {
  switch(action.type){
    case C.ADD_DAY:
    const hasDay = state.some(
      skiDay => skiDay.date === action.payload.date)
      return (hasDay) ?
      state :
      [...state,skiDay(null,action)].sort(
        (a,b) => new Date(b.date) - new Date(a.date)
      )
    case C.REMOVE_DAY: return state.filter(
        skiDay => skiDay.date !== action.payload
      )
    default: state
  }
}
```

۲.۹.۱۴ فایل index.js

```
import C from './constants'
import { allSkiDays } from './store/reducers'

const state = [{
  "resort": "Letmalcan Park",
  "date": "2016-12-6",
  "powder": true,
  "backcountry": false
},{
  "resort": "Letmalcan Mountains",
  "date": "2016-12-10",
  "powder": true,
  "backcountry": true
},
]
const action = {
  type: C.REMOVE_DAY,
  payload: "2016-12-6"
}

const nextState = allSkiDays(state, action)

console.log(`

      initial state: ${JSON.stringify(state)}
      action: ${JSON.stringify(action)}
      new state: ${JSON.stringify(nextState)}

  `)
```

۳.۹.۱۴ خروجی

حالا npm-start استفاده میکنیم

۱۰.۱۴ چالش a

۱.۱۰.۱۴ پیشنیاز

ابتدا به پوشه start در فایل زیپ همین لتک برو محمدامین! محتویاتش را کپی کن بجای محتویات پوشه rx-html انگاه به ترمینال برو و از دستور زیر استفاده کن:

```
npm install
```

۲.۱۰.۱۴ فایل reducers.js

```
export const fetching = (state=false , action) => {
  switch(action.type){
    case C.FETCH_RESORT_NAMES: return true
    default: return state
  }
}
```

۳.۱۰.۱۴ خروجی در ترمینال

```
npm run challenge-a
```



۱۱.۱۴ چالش b

۱.۱۱.۱۴ فایل reducers.js

```
case C.CANCEL_FETCHING: return false
```

۲.۱۱.۱۴ خروجی در ترمینال

```
npm run challenge-b
```

۱۲.۱۴ چالش c

۱.۱۲.۱۴ فایل reducers.js

```
export const suggestions = (state=[] ,action) => {
  switch(action.type){
    case C.CLEAR_SUGGESTIONS: return []
    default: return state
  }
}
```

۲.۱۲.۱۴ خروجی

```
npm run challenge-c
```

۱۳.۱۴ چالش d

۱.۱۳.۱۴ فایل reducers.js

اینجا قطعه کد بزرگی امده که قبلی ها هم باهاشن! فقط دو خط کد اضافه شده که در حقیقت یکی هست و کپی شده با کمی تغییر! 😉 محمدامین میتونی بگی اون یه خط کدومه؟

```
export const fetching = (state=false , action) => {
  switch(action.type){
    case C.FETCH_RESORT_NAMES: return true
    case C.CANCEL_FETCHING: return false
    case C.CHANGE_SUGGESTIONS: return false
    default: return state
  }
}

export const suggestions = (state=[] ,action) => {
  switch(action.type){
    case C.CLEAR_SUGGESTIONS: return []
    case C.CHANGE_SUGGESTIONS: return action.payload
    default: return state
  }
}
```

۲.۱۳.۱۴ خروجی

```
npm run challenge-d
```

۱۴.۱۴ کنار هم اوردن reducer ها

۱.۱۴.۱۴ اگه حال داری بخون!

این ۴ چالش ۴ تا reducer محسوب میشوند و pure-function هستند. همانطور که میدانید pure-function برای برنامه نویسی تابعی یا همان functional-programming استفاده میشه! یه نوع برنامه نویسی دیگه هم یادم میاد به نام OOP که معنیشو همونطور که میدونی میشه object-oriented-programming !

۲.۱۴.۱۴ پیشنیاز

```
npm install redux --save
```

۳.۱۴.۱۴ فایل reducers.js

```
import { combineReducers } from 'redux'
```

در بالای صفحه این را وارد کردیم و حالا به پایین صفحه اضاف میکنیم:

```
export default combineReducers({
  allSkiDays,goal,errors,resortNames: combineReducers({
    fetching,suggestions
  })
})
```

۴.۱۴.۱۴ فایل index.js

```
import C from './constants'
import appReducer from './store/reducers'
import initialState from './initialState.json'

let state = initialState

console.log(`

    InitialState
    ============
    goal: ${state.goal}
    reducers: ${JSON.stringify(state.allSkiDays)}
    fetching: ${state.resortNames.fetching}
    suggestions: ${state.resortNames.suggestions}

  `)

    state = appReducer(state, {
      type: C.SET_GOAL,
      payload: 2
    })

    state = appReducer(state, {
      type: C.ADD_DAY,
      payload: {
        "resort": "Mt Shasta",
        "data": "2016-10-28",
        "powder": false,
        "backcountry": true
      }
    })

  state = appReducer(state, {
    type: C.CHANGE_SUGGESTIONS,
    payload: ["Mt Tallac","Mt Hood","Mt Shasta"]
  })

    console.log(`

        Next state
        ============
        goal: ${state.goal}
        reducers: ${JSON.stringify(state.allSkiDays)}
        fetching: ${state.resortNames.fetching}
        suggestions: ${state.resortNames.suggestions}

      `)
```

۵.۱۴.۱۴ فایل package.json

```
"scripts": {
    "start": "babel-node ./src/"
  },
```

۶.۱۴.۱۴ خروجی

```
npm start
```

فصل ۱۵

استور و اکشن، action store

۱.۱۵ استور store بسازید!

۱.۱.۱۵ پیشنیاز

😀 بابل و وبپک مورد نیاز هستند. برای نصب پکیج هایشان که تعداد نسبتا زیادی دارند این کاری که میگمو بکن! برو پوشه start۳ در همین لتک را به Rx-html بریز! بعد در ترمینال در مسیر همین بزن npm-i انگاه خودش همرو با همون ورژنی که در کانفیگش اومده نصب میکنه! حله داداش!!

۲.۱.۱۵ فایل index.js

```
import C from './constants'
import appReducer from './store/reducers'
import initialState from './initialState.josn'
import {createStore} from 'redux'

const store = createStore(appReducer)
console.log('initialState', store.getState());
store.dispatch({
  type: C.ADD_DAY,
  payload: {
    "resort": "Letmalcan Mountains",
    "date": "2016–10–28",
    "powder": false,
    "backcountry": false
  }
})
console.log('next state', store.getState());
```

۳.۱.۱۵ خروجی

به ترمینال میرویم و در مسیر قرار میگیریم. سپس از npm-start استفاده میشود. بروزر را باز میکنیم به ادرس localhost:۳۰۰۰ میرویم و با دکمه F۱۲ کنسول را باز میکنیم.

۴.۱.۱۵ یک قطعه کد دیگر

کلمه initialState اضاف شده است به قطعه کد زیر:

```
const store = createStore(appReducer, initialState)
```

۵.۱.۱۵ خروجی

کافیست یه نگاه به کنسول بندازیم!

۲.۱۵ متد subscribe

۱.۲.۱۵ فایل index.js

دو کنسول لوگ را حذف کرده، initialState و ایمپورتش را نیز حذف کرده و بجای اینها اضاف میکنیم در قبل dispatch :

```
store.subscribe(() => console.log(store.getState()))
```

۲.۲.۱۵ خروجی

یک ابجکت دربروزر دیده میشود.

۳.۲.۱۵ فایل index.js

به انتهای فایل اضاف:

```
store.dispatch({
  type: C.SET_GOAL,
  payload: 2
})
```

۴.۲.۱۵ خروجی

حالا خروجی دو ابجکت دارد.

۵.۲.۱۵ فایل index.js

به بعد از اولین subscribe اضاف:

```
store.subscribe( () => {
  const state = JSON.stringify(store.getState())
  localStorage['redux-store']=state
} )
```

۶.۲.۱۵ خروجی

در کنسول وارد میکنیم

```
localStorage
```

۷.۲.۱۵ فایل index.js

کل کد هایمان این باشد

```
import C from './constants'
import appReducer from './store/reducers'
import {createStore} from 'redux'

const initialState = (localStorage['redux-store']) ?
JSON.parse(localStorage['redux-store']) : {}

const store = createStore(appReducer, initialState)

window.store = store
store.subscribe( () => {
    const state = JSON.stringify(store.getState())
    localStorage['redux-store'] = state
  } )
```

۸.۲.۱۵ خروجی

در کنسول وارد میکنیم

```
store.getState()
localStorage
localStorage.clear()
localStorage // 0
clear
store.getState() //10 goals
store.dispatch({type: "SET_GOAL", payload: 12})
store.getState()  //12 goals
// clear button
store.getState() //12 goals
```

۳.۱۵ متد Unsubscribe

۱.۳.۱۵ فایل index.js

```
  import C from './constants'
import appReducer from './store/reducers'
import {createStore} from 'redux'

const store = createStore(appReducer)
const unsubscribeGoalLogger = store.subscribe(
  () => console.log(`
  Goal: ${store.getState().goal}`) )
setInterval( () => {store.dispatch({
  type: C.SET_GOAL,payload: Math.floor(
    Math.random()*100 )})}  ,250)
setTimeout(() => {unsubscribeGoalLogger()},3000)
```

۲.۳.۱۵ خروجی

در کنسول تا سه ثانیه اعدادی بصورت رندم میان. اگه بروزر رفرش کنیم دوباره از اول این اتفاق میفته تا سه ثانیه!!

۴.۱۵ ساختن Middleware

چگونگی dispatch شدن action ها توسط Middleware تعیین میشه!

۱.۴.۱۵ فایل store/index.js

```
import C from '../constants'
import appReducer from './reducers'
import { createStore, applyMiddleware } from 'redux'

const consoleMessages = store => next => action => {

  let result

  console.groupCollapsed(`dispatching action => ${action.type}`)
  console.log('ski days', store.getState().allSkiDays.length)
  result = next(action)

  let { allSkiDays, goal, errors, resortNames } = store.getState
      ()

  console.log(`

    ski days: ${allSkiDays.length}
    goal: ${goal}
    fetching: ${resortNames.fetching}
    suggestions: ${resortNames.suggestions}
    errors: ${errors.length}

  `)

  console.groupEnd()

  return result

}

export default (initialState={}) => {
  return applyMiddleware(consoleMessages)(createStore)(
      appReducer, initialState)
}
```

۲.۴.۱۵ فایل index.js

```
import C from './constants'
import storeFactory from './store'

const initialState = (localStorage['redux-store']) ?
  JSON.parse(localStorage['redux-store']) :
  {}

const saveState = () => {
  const state = JSON.stringify(store.getState())
  localStorage['redux-store'] = state
}

const store = storeFactory(initialState)

store.subscribe(saveState)

store.dispatch({
  type: C.ADD_DAY,
  payload: {
    "resort": "Mt Shasta",
    "date": "2016-10-28",
```

```
      "powder": true,
      "backcountry": true
    }
})

store.dispatch({
    type: C.ADD_DAY,
    payload: {
      "resort": "Squaw Valley",
      "date": "2016-3-28",
      "powder": true,
      "backcountry": false
    }
})

store.dispatch({
    type: C.ADD_DAY,
    payload: {
      "resort": "The Canyons",
      "date": "2016-1-2",
      "powder": false,
      "backcountry": true
    }
})
```

۳.۴.۱۵ خروجی

به کنسول میرویم. باید سه تا چیز باشه که روز ها یکی از انچه زیر کشویی ان است کمتر باشد.
رفرش کنید صفحه را همه چی برابر میشه. قبل رفرش میشه روزارو مثلا نام روز را تغییر داد انگاه یکی کمتر میشه.

۵.۱۵ اکشن کریتور actionCreator شماره ۱

اکشن کریتور: actionCreator ها یک تابع هستند که در شکم return انها action برگردانده میشود. ☺ به همین سادگی!

۱.۵.۱۵ پیشنیاز

بعنوان پیشنیاز فایل start۴ و npm-i استفاده شوند.

۲.۵.۱۵ فایل index.js

```
import storeFactory from './store'
import {addDay} from './actions'

const store = storeFactory()

store.dispatch(addDay('Letmalcan','98.07.20'))
```

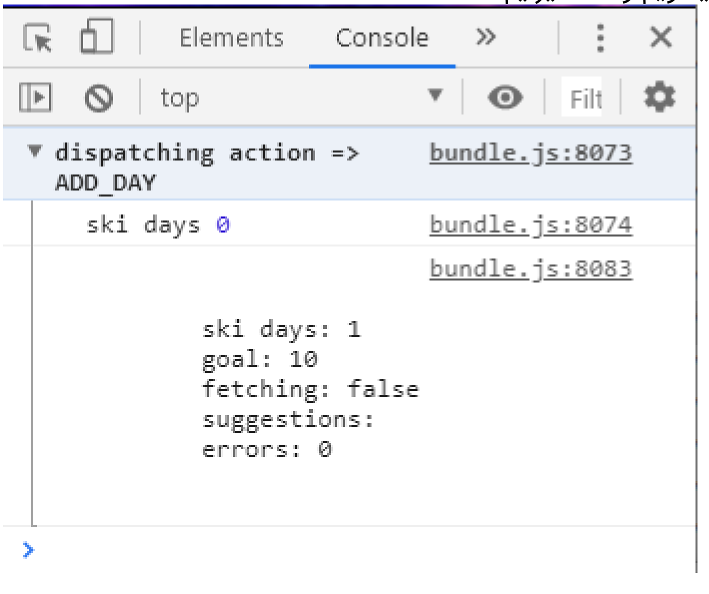
۳.۵.۱۵ فایل actions.js

اضافه میکنیم. این تابع یک actionCreator است و در ادامه چنتا دیگه actionCreator بیشتر خواهیم ساخت. خیلی خوب یاد میگیریم. داره تکرار میشه به حافظه میره حتی اگه نخوایم یاد بگیریم.

```
  export function addDay(resort,date,powder=false,backcountry=
      false){
  return{
    type: C.ADD_DAY,
    payload:{resort,date,powder,backcountry}
  }
}
```

۴.۵.۱۵ خروجی

به ترمینال میریم و npm-start وارد میکنیم سپس وارد بروزر به ادرس http://localhost:۳۰۰۰/ میشویم و F۱۲ میزنیم.



۶.۱۵ اکشن کریتور شماره ۲

۱.۶.۱۵ فایل actions.js

```
 export const removeDay = function (date){
  return{
    type: C.REMOVE_DAY,
    payload: date
  }
}

export const setGoal = function(goal){
  return {
    type: C.SET_GOAL,
    payload: goal
  }
```

۲.۶.۱۵ فایل index.js

حالا همون دو تا اکشن کریتور که الان درست کردیم اینجا واردشون میکنیم! فک میکنم با این کدهای خوب مطالب آسونی داشته باشه ریداکس!! درست میگم محمدامین؟

```
import {addDay, removeDay, setGoal} from './actions'
// ... another codes
store.dispatch(removeDay("98.07.20"))
store.dispatch(setGoal(55))
```

۳.۶.۱۵ خروجی

لازم نیست اینبار مراحل را دوباره برم، کافیه فقط یه نگاه به بروزر بندازم!

۷.۱۵ اکشن کریتور های بعدی

۱.۷.۱۵ پیشنیاز

در این بخش به ترمینال بروید تا یک پکیج جدید به نام expect نصب کنیم. برای اینکه از حالت اتوماتیک قبلی خارج شود از ctrl+c و در جواب از n استفاده کنید و اینتر بزنید! حالا نصب کنید

```
npm install expect --save
```

۲.۷.۱۵ فایل actions.js

اضاف بشهبه قبلیا، لطفا !!

```
export const addError = (message) => {
  return ({
    type: C.ADD_ERROR,
    payload: message
  }) }
export const clearError = index => {
  return ({
    type: C.CLEAR_ERROR,
    payload: index
  }) }
export const changeSuggestions = x => {
  return ({
    type: C.CHANGE_SUGGESTIONS,
    payload: x
  }) }
export const clearSuggestions = () => {
  return ({ type: C.CLEAR_SUGGESTIONS }) }
```

۳.۷.۱۵ فایل index.js

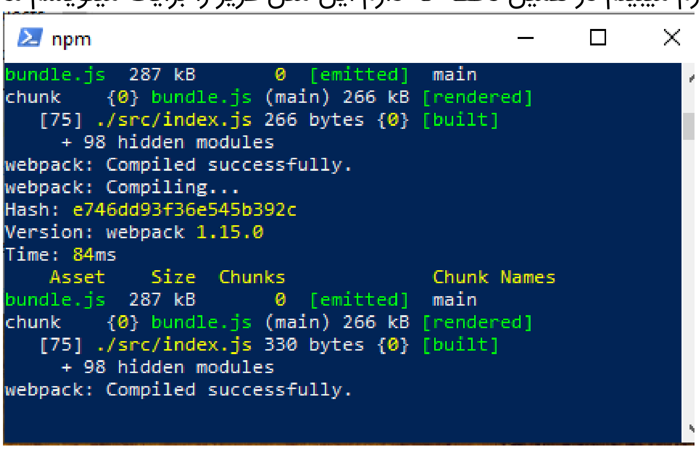
```
import expect from 'expect'
import storeFactory from './store'
import {addError,clearError,changeSuggestions,clearSuggestions}
    from './actions'

const store = storeFactory()
//1
store.dispatch(addError("sth went wrong!!"))
expect(store.getState().errors).toEqual(["sth went wrong!!"])
console.log(`addError actionCreator successfully worked!!`)
//2
store.dispatch(clearError(o))
expect(store.getState().errors).toEqual([])
console.log(`successfully worked!!`)
//3
store.dispatch(changeSuggestions(['one','two','three']))
expect(store.getState().resortNames.suggestions).toEqual(['one',
    'two','three'])
console.log(`successfully worked!!`)
//4
store.dispatch(clearSuggestions())
expect(store.getState().resortNames.suggestions).toEqual([])
console.log(`successfully worked!!`)
```

۴.۷.۱۵ خروجی

از npm-start استفاده میکنمو به کنسول بروزر میرم!

😮 این تصویرو ببینید!! اینو گذاشتم که فقط ببینی محمدامین که چه تصاویر قشنگی من الان دارم میبینم در همین لحظه که دارم این متن عزیز را برایت مینویسم محمدامین!!



۸.۱۵ تانک thunk

۱.۸.۱۵ پیشنیاز

نصب یک پکیج را به همراه داریم عزیزان همه با هم این پکیج را نصب میکنیم!! به یاد همه اونایيکه میخواستن امروز برن کوهنوردی و تا قله موفق به پیشروی شدن!

```
npm i redux-thunk --save
```

۲.۸.۱۵ فایل store/index.js

به داخل پوشه store میرویم و در فایل index وارد میشود:

```
import thunk from 'redux-thunk'
```

در پایین ترین خط، کلمه thunk را وارد میکنیم:

```
return applyMiddleware(thunk,
```

۳.۸.۱۵ فایل actions.js

```
export const randomGoals = () => (dispatch,getState) => {
  if(!getState().resortNames.fetching){
    dispatch({type: C.FETCH_RESORT_NAMES})
    setTimeout(() => {dispatch({type: C.CANCEL_FETCHING})}
        ,1500)
  }}
```

این تانک، STATE موجود را بررسی میکند.

۴.۸.۱۵ فایل index.js

```
import storeFactory from './store'
import {randomGoals} from './actions'

const store = storeFactory()

store.dispatch(randomGoals())
```

۵.۸.۱۵ خروجی

یه npm-start بعد نیگا کن کنسول بروزرو!! دوتا هست! فقط یکی در index نوشتیم اما اینجا دوتا نشون میده! تانکه دیگه! البته خیلی چیزاشو خودش در فایل نوشته بود و نفهمیدم! همون بهتر که نفهمیدم! کار من کپ زدنه نه یادگیری فعلا!

۹.۱۵ سرور و autocomplete-thunk

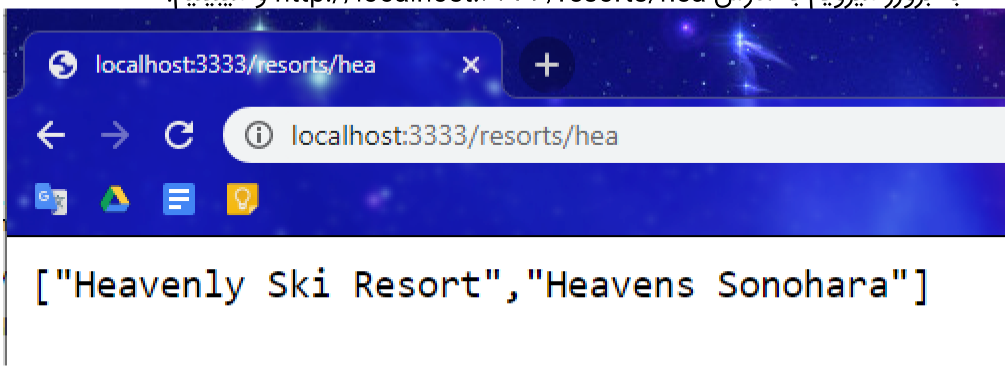
۱.۹.۱۵ پیشنیاز

پوشه start۵ را استفاده میکنیم با npm-i در مسیر مخصوصش که معمولا همراهش از npm-audit-fix هم استفاده میکنم.

```
npm i isomorphic-fetch --save
npm run suggestions
```

البته بیارم بدون نصب این پکیج isomorphic-fetch از فتچ استفاده کردم جواب داد و با ایمپورت کردن پکیج ارور میداد. در ورژن های جدیدی اینطوره ولی اینجا لیندا ما قدیمیه و همین کدها جواب میده!

به بروزر میرویم به ادرس http://localhost:۳۳۳۳/resorts/hea و میبینیم:



۲.۹.۱۵ فایل index.js

کلا همینه!

```
import storeFactory from './store'
import {suggestResortNames} from './actions'
const store = storeFactory()
store.dispatch(suggestResortNames("hea"))
```

۳.۹.۱۵ فایل actions.js

به پایینش اضافه میکنیم:

```
  export const suggestResortNames = value => dispatch => {
      dispatch({type: C.FETCH_RESORT_NAMES})
      fetch('http://localhost:3333/resorts/' + value)
          .then(response => response.json())
          .then(suggestions => {dispatch({
                  type: C.CHANGE_SUGGESTIONS,
                  payload: suggestions
              }) })
          .catch(error => {
              dispatch(addError(error.message))
              dispatch({type: C.CANCEL_FETCHING})
          }) }
```

۴.۹.۱۵ خروجی

یک ترمینال دیگر باز میکنیم و npm-start میزنیم. یک بروزر دیگه هم به ادرس ۳۰۰۰ میرویم به کنسول! الان خوب به سرور وصل میشه! حالا اون ترمینال اولی را با ctrl+c قطع میکنیم و به کنسول نگاه میندازیم ارور میده! همچنین ارور ان در دوتا پایینی نمایش داده میشه! روبروی ارور مینوسه ۱.

۱۰.۱۵ متصل کردن mapping

۱.۱۰.۱۵ پیشنیاز

پوشه start۶ را استفاده میکنیم با npm-i در مسیر مخصوصش که معمولا همراهش از npm-audit-fix هم استفاده میکنم. راستش اینکارو نکردم. فقط کدارو نوشتم. حالا پوشه بندی را از start۶ میبینم. بخاطر قدیمی بودنش اینکارو نتونستم انجام بدم.

```
npm i react-redux --save
npm run suggestions
//new terminal
npm start
```

۲.۱۰.۱۵ فایل index.js

```
import sampleData from './initialState'
import storeFactory from './store'
import {Provider} from 'react-redux'

const initialState = (localStorage["redux-store"]) ?
    JSON.parse(localStorage["redux-store"]) :
    sampleData

const saveState = () =>
    localStorage["redux-store"] = JSON.stringify(store.getState
        ())

const store = storeFactory(initialState)
store.subscribe(saveState)

window.React = React
window.store = store

render( <Provider store={store}> {routes} </Provider>
  ,document.getElementById('react-container')
)
```

۳.۱۰.۱۵ خروجی

در inspect-element به قسمت کنسول برو!

```
localStorage.clear()
store.getState()
// an object come from initialState.json
```

😀 از بخش کنسول به بخش react که پلاگینشو روی کروم نصب کردیم برو! یک اسکرین شات بگیر! از تگ های ری اکت!! قشنگن مگه نه؟

۱۱.۱۵ مپ شماره ۲

۱.۱۱.۱۵ فایل SkiDayCount.js

یه فایل هست به در مسیر src/components/containers/SkiDayCount.js
در کنار همین پوشه یک پوشه به نام ui وجود دارد که فایل های درونش دقیقا همین نام های فایل های این پوشه را دارند. بنابراین از پوشه containers اطلاعات به پوشه ui میرود.

```
import SkiDayCount from '../ui/SkiDayCount'
import {connect} from 'react-redux'

const mapStateToProps = state => {return{
  total: state.allSkiDays.length,
  powder: state.allSkiDays.filter(
    day => day.powder
  ).length,
```

```
    backcountry: state.allSkiDays.filter(
      day => day.backcountry
    ).length
}}
const Container = connect(mapStateToProps)(SkiDayCount)
export default Container
```

۲.۱۱.۱۵ خروجی

همچنان ان دو ترمینال فعال هستند پس به بروزر میرویم. نشان میدهد سه اسکی دی را! در بروزر هم سه ابجکت را نشان میدهد.

۱۲.۱۵ مپ شماره ۳

map dispatch to react component

۱.۱۲.۱۵ فایل ShowErrors.js

```
import ShowErrors from '../ui/ShowErrors'
import {clearError} from '../../actions'
import {connect} from 'react-redux'

const mapStateToProps = state => {return{
  errors: state.errors
}}
const mapDispatchToProps = dispatch => {return{
  onClearErrors(index){dispatch(clearError(index))}
}}
export default connect (mapStateToProps,mapDispatchToProps)(
    ShowErrors)
```

۲.۱۲.۱۵ خروجی

به کنسول میرویم و تایپ میکنیم:

```
store.getState()
store.dispatch({type: "ADD_ERROR", payload: "sth went wrong!!"})
```

در بروزر ارور پاپ اپ میشود. در کنسول هم یک پیام میاد که با بستن ارور اپلیکیشن تبدیل میشه به CLEAR_ERROR

۳.۱۲.۱۵ فایل index.js

```
import sampleData from './initialState'
import storeFactory from './store'
import {Provider} from 'react-redux'
import {addError} from './actions'

const initialState = (localStorage["redux-store"]) ?
    JSON.parse(localStorage["redux-store"]) :
    sampleData

const saveState = () =>
    localStorage["redux-store"] = JSON.stringify(store.getState
        ())

const store = storeFactory(initialState)
store.subscribe(saveState)

const handleError = error => {
  store.dispatch(addError(error.message))
}

```

```
window.React = React
window.store = store
window.addEventListener("error",handleError)

render( <Provider store={store}> {routes} </Provider>
  ,document.getElementById('react-container')
)

foo=bar
```

۴.۱۲.۱۵ خروجی

بخاطر خط اخر ارور میده. ارور که میبندیم. در کنسول CLEAR_ERROR نشون میده!

۱۳.۱۵ مپ شماره ۴

۱.۱۳.۱۵ فایل SkiDayList.js

```
import SkiDayList from '../ui/SkiDayList'
import {connect} from 'react-redux'
import {removeDay} from '../../actions'

const mapStateToProps = (state,props) => ({
  days: state.allSkiDays,
  filter: props.params.filter
})
const mapDispatchToProps = dispatch => ({
  onRemoveDay(date){dispatch(removeDay(date))}
})

export default connect(mapStateToProps,mapDispatchToProps)(
    SkiDayList)
```

۲.۱۳.۱۵ خروجی

به صفحه لیست میرویم و میبینیم با دبل کلیک حذف میشه!

۱۴.۱۵ مپ شماره ۵

۱.۱۴.۱۵ فایل GoalProgress.js

```
import GoalProgress from '../ui/GoalProgress'
import {connect} from 'react-redux'
import {setGoal} from '../../actions'

const mapStateToProps = state => ({
  current: state.allSkiDays.length,
  goal: state.goal
})
const mapDispatchToProps = dispatch => ({
  onNewGoal(goal){dispatch(setGoal(goal))}
})
export default connect(mapStateToProps,mapDispatchToProps)(
    GoalProgress)
```

۲.۱۴.۱۵ خروجی

به صفحه اصلی اپ قسمت goal با کلیک تعدادش عوض میشه!

۱۵.۱۵ مپ شماره ۶

۱.۱۵.۱۵ فایل addDayForm.js

```
import AddDayForm from '../ui/AddDayForm'
import {withRouter} from 'react-router'
import {connect} from 'react-redux'
import {AddDay, suggestResortNames, clearSuggestions} from '
    ../../actions'

const mapStateToProps = (state,props) => ({
  suggestions: state.resortNames.suggestions,
  fetching: state.resortNames.fetching,
  router: props.router
})
const mapDispatchToProps = dispatch => ({
  onNewDay({resort,date,powder,backcountry}){
    dispatch(addDay(resort,date,powder,backcountry))
  } ,
  onChange(value){
    if(value){dispatch(suggestResortNames(values))
    }else{dispatch(clearSuggestions())}
  },
  onClear(){dispatch(clearSuggestions())}
})
const Container = connect(mapStateToProps,mapDispatchToProps)(
    AddDayForm)
export default withRouter(Container)
```

۲.۱۵.۱۵ خروجی

بخشی از اپلیکیشن که به روز ها اضافه میشود میرویم و یک روز اضاف میکنیم.

😃جشن اختتامیه! تموم شد! فصل ریداکس با موفقیت خلاصه نویسی شد محمدامین عزیز! هوررااا...



فصل ۱۶

پیش نیاز هر برنامه نویس

16.1 Git

- https://m.youtube.com/watch?v=SWYqp7iY_Tc

Download: https://git-scm.com/download/win

16.2 Linux

- https://git.ir/pluralsight-getting-started-with-linux/
- https://m.youtube.com/watch?v=SWYqp7iY_Tc

فصل ۱۷

نقشه راه

https://roadmap.sh/

فصل ۱۸

استایل

Css preprocessor => Sass/Less/stylus
Css framework => foundation/bootstrap/material-design-bootstrap)
Css architecture => BEM

18.1 Sass

https://sass-lang.com/

18.2 BEM

http://getbem.com/

فصل ۱۹

جمع اوری داده ها در pdf

19.1 Reactjs

https://create-react-app.dev/
https://reacttraining.com/
https://reactjs.org

- React-Router-v4: https://courseclub.me/tylermcginnis-react-rou
- React-and-js-2parts:
  part 01: https://git.ir/skillshare-the-full-javascript-es6-t
  part 02: https://git.ir/skillshare-the-full-javascript-es6-t
- js-full: https://git.ir/skillshare-complete-javascript-course
- https://git.ir/skillshare-make-youtube-application-with-r
- https://git.ir/packtpub-react-front-to-back/

19.2 Redux

https://redux-starter-kit.js.org/
https://react-redux.js.org/
https://redux.js.org/

- Redux-msod
- Redux-Reactjs
- Redux-lynda

19.3 React.Native

main source: https://facebook.github.io/react-native/

- alex banx https://p30download.com/fa/entry/84718/

هرموقع لپتاپ با بهتر از اون مک، خریدم برمیگردم اپ دولوپ میکنم با نیتیو.
http://bitdownload.ir/lynda-react-native-essential-training/ samoe
https://download.ir/ steve

فصل ۲۰

بک اند با js

20.1 PWA

https://git.ir/packtpub-learn-to-build-progressive-web-apps-u

20.2 SQL

- SQL beginner: https://git.ir/pluralsight-getting-started-with
- SQL full: https://git.ir/packtpub-the-complete-mysql-develop